

HPE Virtual User Generator

Software Version: 12.55

User Guide

Go to **HELP CENTER ONLINE**

<http://vugenhelp.saas.hpe.com>



Hewlett Packard
Enterprise

Document Release Date: July 2017 | Software Release Date: August 2017

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 1993-2017 Hewlett Packard Enterprise Development LP

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Contents

HPE Virtual User Generator	1
Wellcome to the VuGen User Guide	28
What's New	28
New - MQTT Protocol	28
New - support for JMeter tests	28
New supported technologies and platforms	29
Protocol enhancements	29
TruClient enhancements	31
VuGen enhancements	31
Controller and Analysis enhancements	32
Network Virtualization enhancements	32
Integration enhancements	33
VuGen	34
Introducing VuGen	34
Overview	34
Use VuGen scripts to test your environment	34
VuGen Updates	35
Vusers	35
Vuser Technology	35
Vuser Types	36
VuGen User Interface	37
VuGen Workspace	37
How to Modify the VuGen Layout	40
Keyboard Shortcuts	42
File Menu	42
Edit Menu	43
View Menu	43
Search Menu	44
Design Menu	44
Record Menu	45
Replay Menu	45
ALM Menu	45
Windows Menu	46
Solution Explorer Pane	46
Understanding the Solution Explorer	47
Solution Explorer Structure and Context Menu Options	49
Step Navigator Pane	52
Editor Pane	54

Supported Programming Languages	55
Tips for Working with the Editor	56
Code Completion and Tooltips	56
Code-Coloring	57
Script Folding	58
Community Search	58
User Interface Elements	58
Steps Toolbox Pane	60
Bookmarks Pane	61
Snapshot Pane	62
Standard Snapshot pane controls	63
Snapshot pane controls for Citrix, RDP, and SAP protocols	64
Snapshot pane controls for Windows Sockets protocol	66
Snapshot pane controls for Web - HTTP/HTML protocol	66
Snapshot pane controls for protocols with an XML request/response (such as Web Services)	68
Snapshot pane controls for Database Protocols	69
Copying images to the clipboard	70
Copying snapshot text to the clipboard	70
Customized Snapshot pane functionality	70
Activating the Snapshot on error functionality	71
Thumbnail Explorer	71
Errors Pane	72
Tasks Pane	74
Task Editor	75
Output Pane	77
Breakpoints Pane	78
Call Stack Pane	81
Watch Pane	81
Runtime Data Pane	82
.NET Recording Filter Pane	83
Options Dialog Box	86
General Options Tab	86
Editor Options Tab	90
Scripting Options Tab	96
Search and Replace Dialog Boxes	105
Business Process Report Dialog Box	106
Replay Summary Pane	108
VuGen Workflow	113
Creating or Opening Vuser Scripts	113
Creating Vuser Scripts - Overview	113
Create and Open Vuser Scripts	114
Compare Scripts Side-by-Side	115
Working with Application Lifecycle Management	115

Managing Scripts Using ALM - Overview	115
Connect to ALM	115
ALM Version Control - Overview	116
Work with Scripts in ALM Projects	116
Work with Version-Controlled Scripts in ALM Projects	117
Save VuGen Vuser Scripts to ALM Projects	118
Compare Previous Versions of a Script	118
HPE ALM Connection Dialog Box [VuGen]	119
Working with Git	120
Managing Scripts with Git - Overview	120
Work with Scripts in Git	121
Troubleshooting	121
Git Operations	122
Configure Ignore List Dialog Box	124
Resolve Conflicts Dialog Box	125
Import from Remote Repository Dialog Box	126
Multiple Protocol Scripts	126
Script Directory Files	127
Create a New Script Dialog Box	128
How to Work with .zip Files	129
Import from a Zip File	129
Export to a Zip File	130
Zip and Email	130
Edit Script in Zip File	130
Create and Open Vuser Script Templates	130
Create a Vuser Script Template	130
Create a Vuser Script From a Template	130
Rename a Vuser Script Template	130
Template Guidelines	131
Vuser Script Templates	131
Template Guidelines	131
Recording	131
Recording - Overview	131
Vuser Script Sections	132
VuGen Script Editor	133
Java Classes	133
Script Section Structure Example	133
Header Files	134
Record a Vuser Script	135
Recording Summary Report	137
Enable the Recording Summary report	137
Access the Recording Summary report	137
Recording Summary report overview	137

Filter recorded script using the Recording Summary report	139
Regenerate the script	140
Create a Business Process Report	140
Types of information included in business process reports	140
Create a business process report	141
Configure additional report options	141
Recording Options	141
Citrix > Configuration Recording Options	141
Citrix > Code Generation Recording Options	142
Citrix > Login Recording Options	142
ICA File Structure	144
Citrix > Recorder - Recording Options	145
COM/DCOM > Filter Recording Options	146
COM/DCOM > Options Recording Options	148
Correlations > Configuration Recording Options	149
Correlations > Rules Recording Options	151
Advanced Correlation Properties Dialog Box	154
Token Substitution Testpad Dialog Box	155
Database > Database Recording Options	156
Database > Advanced Recording Options Dialog Box	156
Data Format Extension > Chain Configuration Recording Options	157
Add Prefix/Postfix to Chain Dialog Box	158
Add Data Format Extension	160
Data Format Extension > Code Generation Recording Options	161
Flex > RTMP Recording Options	162
Flex > Configuration Recording Options	163
Flex > Externalizable Objects Recording Options	163
FTP > Configuration Recording Options	165
General > Code Generation Recording Options	165
General > Protocol Recording Options	166
General > Recording - Recording Options	166
Advanced URL Dialog Box	167
Advanced HTML Dialog Box	168
General > Script Recording Options	169
GUI Properties > Web Event Configuration Recording Options	172
Custom Web Event Recording Configuration Dialog Box	172
GUI Properties > Advanced Recording Options	174
HTTP Properties > Advanced Recording Options	175
Headers Dialog Box	179
Content Type Filters Dialog Box	180
Non-Resources Dialog Box	181
Java > VM Recording Options	181
Java > Classpath Recording Options	182

Microsoft .NET > Recording - Recording Options	182
Remote Objects Property	184
Network > Mapping and Filtering Recording Options	185
Server Entry - Port Mapping Dialog Box	187
Advanced Port Mapping Settings Dialog Box	189
Server Entry - Traffic Filtering Dialog Box	190
RDP > Code Generation > Advanced Recording Options	191
RDP > Code Generation > Agent Recording Options	191
RDP > Code Generation > Basic Recording Options	192
RDP > Client Startup Recording Options	194
Recording Properties > Corba Options Recording Options	194
Recording Properties > Correlation Options - Recording Options	195
Recording Properties > Log Options Recording Options	196
Recording Properties > Recorder Options - Recording Options	197
Recording Properties > Serialization Options - Recording Options	198
RTE > Configuration Recording Options	199
RTE > RTE Recording Options	199
SAPGUI > Auto Logon Recording Options	200
SAPGUI > Code Generation Recording Options	201
SAPGUI > General Recording Options	201
Traffic Analysis > Traffic Filters Recording Options	202
WinSock Recording Options	203
Recording Options - Miscellaneous Topics	204
Protocol Compatibility Table	204
Port Mapping and Traffic Filtering Overview	207
Port Mapping Auto Detection	208
EUC-Encoding (Japanese Windows only)	209
Script Generation Preference Overview	210
Script Language Options	210
Recording Levels - Overview	211
Serialization Overview	212
Tips for Working with Event Listening and Recording	212
Providing Authentication Information for Multi-Protocol Scripts	213
How Internet Explorer authenticates users	213
Generating a web_set_user function	214
Recording via a Proxy - Overview	214
Record a Script via a Proxy	216
Use Case 1	216
Use Case 2	217
Use Case 3	218
Use Case 4	219
After recording	219
Import Actions to a Script	219

Regenerate a Vuser Script	220
Start Recording Dialog Box	221
Floating Recording Toolbar	225
Files Generated During Recording	227
Troubleshooting and Limitations for Recording with VuGen	229
Proxy recording	229
Security Levels	229
Troubleshooting missing steps	229
Recording on Internet Explorer 10	230
Recording on Microsoft Edge	230
Certificate warning message	230
Multi-Protocol recording	230
Overwriting of data	231
Firefox as default browser	231
FTP and Active SSL	231
HSTS Web Recording	231
FTP Recording	231
64-bit Recording	231
Correlating	232
Correlation Overview	232
Correlation Tab [Design Studio] Overview	234
Automatic Correlation	236
Rule-based correlation	236
Recording-based correlations	237
Replay-based correlations	237
Automatic Correlation Configuration	237
Exclude Strings or Content Types from the Correlation Scan	241
Design Studio	241
Manually Correlate Scripts	242
Design Studio [Correlation Tab] Dialog Box	242
Modify Correlation Definitions	246
Modifying Boundary Based Correlation Definitions	246
Modifying Regular Expression Correlation Definitions	247
Modifying XPath Correlation Definitions	247
Modifying Winsocket Correlation Definitions	247
Correlate Scripts Using Design Studio	250
Advanced Correlation Techniques	251
Search for Values that Need Correlation	251
Search by Comparing Scripts	251
Replay Log Search	252
Wdiff Correlation Utility	252
Modifying Saved Parameters	253
Correlation Specifics for Protocols	253

COM Correlation	253
C Vuser Scripts Correlation Functions	253
Database, RTMP and COM Correlation from a Snapshot	254
Flex (RTMP/AMF) Correlation	255
Flex XPath Correlation	258
Java Custom Correlation	258
Java Scripts Correlation - Serialization	260
Microsoft .NET Correlation	263
Oracle NCA Correlation	264
Correlate Statements for Load Balancing	264
Correlate the JServSessionIdroot Values	265
Siebel Correlation	266
Web Manual Correlations	271
Web Manual Correlation in Code	272
Web-based Correlation Using Design Studio	273
Winsock Correlation	275
Winsock Manual Correlation	276
Replaying	277
Replay Overview	277
Replay a Vuser Script	278
Check Linux Compatibility	278
Work with Snapshots	279
Show the Snapshot pane	279
Copy a snapshot to the clipboard	280
Copy snapshot text to the clipboard	280
Activate the snapshot-on-error functionality	280
Set the snapshot options	281
Troubleshooting Snapshots	281
Snapshots that Have an XML View	281
Add a Text Check From the XML View in the Snapshot Pane	282
Running a Vuser as a Process or Thread	283
Runtime Settings	283
Runtime Settings Overview	283
Runtime Setting Value Validation	284
Runtime Settings Views	285
Access Runtime Settings View	285
Runtime Settings View Descriptions	286
Preferences View - Internet Protocol	291
Defining Non-Critical Resources	303
Prevent unnecessary failures due to errors with non-critical resources	303
Resource classification	304
Example	304
Import and Export Runtime Settings	304

Export runtime settings	304
Import runtime settings	305
Revert runtime settings to the default settings	305
Configure Runtime Settings Manually	305
Bookmarks Overview	308
Run a Vuser Script from a Command Prompt	308
Use Bookmarks	309
Create a Bookmark	310
Remove a Bookmark	310
Navigate between Bookmarks	310
Navigate to a Specific Bookmark in a Vuser Script	310
Files Generated During Replay	310
NV Insights Report	312
Opening the NV Insights Report	312
Limitations	313
Debugging	313
Debugging Overview	313
Running a Vuser script	313
The Run Step by Step Command	314
Breakpoints	314
Bookmarks	314
Watching Variables	315
Go To Commands	315
Output Pane	315
Correlation	315
Error Handling	315
Additional Debugging Information	316
General Debugging Tip	316
Using C Functions for Tracing	316
Additional C Language Keywords	316
Add Additional Keywords	317
Examining Replay Output	317
Working with Breakpoints	317
Breakpoints Pane	318
Watching Expressions and Variables	318
Debugging Web Vuser Scripts	320
Debug Scripts with Breakpoints	320
Enhancing	321
Enhancing a Script for Load Testing - Overview	321
Add Parameterization	321
Insert Transactions	322
Insert Rendezvous Points	322
Insert VuGen Functions	322

Insert Steps	322
Insert Comments	322
Insert Log Messages	323
Insert Synchronization Points (RTE Vusers only)	323
Transaction Overview	323
Insert Transactions	324
Insert a transaction while recording	324
Insert a transaction after recording	324
Transaction guidelines	325
Display Transactions	326
Cross-Vuser Transaction Overview	326
Create a Cross-Vuser Transaction	327
Rendezvous Points	328
Adding VuGen Functions Overview	328
Obtain Vuser Information	328
Send Messages to Output	329
General Vuser Functions	330
Protocol-Specific Vuser Functions	330
Encoding Passwords and Text	331
Password Encoder	332
Masking Text	333
Database Integration Overview	334
Connecting to a Database	334
Using Data Retrieved from SQL Queries	335
Validating Database Values	337
Checking Returned Values Through a Database	338
Performing Actions on Datasets	339
Create a Controller Scenario from VuGen	340
Insert Steps into a Script	340
Insert Think Time Steps	340
Insert Debug Messages	340
Insert Error and Output Messages	341
Create Controller Scenario Dialog Box	341
Parameterization	342
Parameterization Overview	342
Delimiters	342
Input/Output parameters	343
Correlation	344
VTS and Parameterization	344
Parameter Types	345
File Type Parameters	345
Table Type Parameters	345
XML Type Parameters	345

Internal Data Type Parameters	346
User-Defined Function Parameters	346
Create Parameters	347
Import Parameter Values from a File	348
Work with Existing Parameters	348
Data Assignment Methods for File-Type Parameters	349
Data Assignment and Update Methods for File Parameters	350
Vuser Behavior in the LoadRunner Controller	351
XML Parameters	352
Create an XML Parameter from a Web Service Call	352
Create XML Parameters - Standard Method	353
Define XML Value Sets	354
Set an Assignment Method	357
Modify XML Parameter Properties	358
Set AUT Environment Parameters	358
Select or Create Parameter Dialog Box	359
Parameter Properties Dialog Box	359
Date/Time, Group Name, Iteration Number, Load Generation Name, and Vuser ID Parameters	360
File Parameters	362
Table Parameters	364
Random Number Parameters	366
Unique Number Parameters	367
User Defined Function Parameters	368
XML Parameters	369
Parameter Simulation Dialog Box	369
Parameter List Dialog Box	373
Create Parameter Dialog Box	373
Parameter Original Value Dialog Box	374
Parameter Delimiters Configuration Dialog Box	374
Troubleshooting and Limitations for Parameterization	375
Asynchronous Communication	377
Synchronous and Asynchronous Concepts	377
Types of Asynchronous Communication	378
VuGen Support for Asynchronous Communication	380
Create an Asynchronous Vuser Script	381
Create a new Vuser script	382
Enable Async Scan	382
Record the business processes using the typical VuGen recording process	382
Generate, scan, and modify the Vuser script	382
Review the modifications that VuGen made to the script	382
Asynchronous Communication API	383
How Asynchronous Functions Differ from Synchronous Functions	384
How VuGen Modifies a Vuser Script for Asynchronous Communication	385

How VuGen modifies Vuser scripts	385
How VuGen modifies flex_amf_call steps	387
Defining the Start of an Asynchronous Conversation	388
Defining the End of an Asynchronous Conversation	389
Using Asynchronous Request Thresholds	391
Fine-Tuning the End of an Asynchronous Conversation	391
Correlating Asynchronous Vuser Scripts	392
Implementing Callbacks	392
Modifying Callbacks	395
Parsing URLs	399
Async Rules Overview	402
Adding Async Rules	402
Adding a positive Async rule	403
Adding a negative Async rule	404
Async Tab [Design Studio]	404
Asynchronous Options Dialog Box	406
Asynchronous Example - Poll	407
Asynchronous Example - Push	411
Asynchronous Example - Long-Poll	413
Viewing Replay Results	415
VuGen Protocols	417
IPv6 Support	420
IPv6 Deployment	420
Protocols Supported	421
Protocol Support Limitations	421
Troubleshooting	421
Protocol Support for Async, IPv6, and 64-bit Recording	422
Ajax - Click & Script Protocol	424
Ajax (Click & Script) Protocol Overview	424
Ajax (Click & Script) Supported Frameworks	425
Ajax (Click & Script) Example Script	426
Ajax (Click & Script) Recording Tips	427
Ajax (Click & Script) - Replay Tips	429
Ajax (Click & Script) Miscellaneous Tips	430
Ajax Click & Script Troubleshooting and Limitations	431
Citrix Protocol	435
Citrix Protocol - Overview	435
Set Up Your Citrix Environment	436
Agent for Citrix Presentation Server - Overview	438
Citrix Recording Tips	441
Citrix Synchronization	443
Citrix - Automatic Synchronization	444
Sync on Window	444

Sync on Obj Info	444
Sync on Text	445
Citrix - Manual Synchronization	446
Citrix - Additional Ways to Synchronize Your Script	446
Failed Bitmap Synchronization Dialog Box	448
Citrix Replaying Tips	448
Wildcards	448
Run as a Process—not a Service	449
Modify the window size of the Citrix client during replay	449
Enable Think Time	449
Set Initialization Quota	449
Set a Bitmap Polling Delay	450
Use Exact Tolerance	450
Set Consistency Between Machines	450
Increasing the Number of Vusers per Load Generator Machine	450
Citrix Debugging Tips	450
Citrix - Troubleshooting and Limitations	452
General Limitations	452
Effects and Memory Requirements of Citrix Agent	454
Random Failures of Functions Accessing Citrix Agent	454
Citrix Agent will not start	454
Unexpected Disconnection	455
Citrix Receiver—Security Warning	455
Failed to get session from client	456
Citrix Error 13 "Unsupported Function"	456
Citrix Error 70, Client Error 1030 "Protocol driver error"	456
Capturing Empty Text	456
Click & Script Protocols	457
Click & Script Protocols - Overview	457
Click & Script Recording Tips	459
Click & Script - Replay Tips	460
Click & Script Miscellaneous Tips	461
Click & Script Enhancements	462
Click & Script API Notes	465
Regular expressions	465
Ordinals	465
Empty strings	466
Click & Script Troubleshooting and Limitations	466
COM/DCOM Protocol	470
COM/DCOM Protocol Overview	470
COM/DCOM Technology Overview	470
Objects, Interfaces, and Type Libraries	471
COM Interfaces	471

COM Class Context and Location Transparency	471
COM Data Types	471
COM/DCOM Vuser Script Structure	471
COM Sample Vuser Scripts	473
Selecting COM Objects to Record	477
Database Protocols	478
Database Protocols Overview	478
VuGen Database Recording Technology	479
Database Grids	480
Correlate a value in a data grid	480
Save the data in a data grid to a file	480
Copy the text from a cell in a data grid to the clipboard	480
Search for data inside a data grid	480
Handling Database Errors	481
Globally Modifying Error Handling	481
Locally Modifying Error Handling	482
Debugging Database Applications	482
Database Protocols - Troubleshooting and Limitations	484
Flex (RTMP/AMF) Protocol	493
Flex Overview	493
What is Flex?	493
Flex Technologies	493
Learn more about developing a Flex Vuser script	494
Recording Flex Scripts	495
AMF	495
RTMP Functions	497
RTMP Tunneled Functions	497
RTMP/RTMPT Streaming	498
RTMP Tunneled	505
Record a Flex Script	506
Setting the Flex Recording Mode	507
Example	507
Code Generation in the Flex Protocol	508
Externalizable Objects in Flex Scripts	509
Flex Correlations	511
Flex Snapshots	511
Serialize Flex Scripts	512
Query an XML Tree	513
Troubleshooting and Limitations for Flex	515
GraniteDS (Data Services)	516
Java Record Replay Protocol	516
Java Record Replay Protocol Overview	516
Supported Java Communication Protocols	516

Before Recording a JRR Script	517
Notes and limitations	518
Java Record Replay Protocol Recording Tips	518
Working with RMI	519
Working with CORBA	520
CORBA Application Vendor Classes	520
Editing a CORBA Vuser Script	521
Working with Jacada	522
Recording a Jacada Vuser	522
Editing a Jacada Vuser Script	523
Working with JDBC	524
Supported Databases	524
Recording JDBC Scripts	524
Limitations	524
Sample Script	524
Manually Insert Java Methods	526
To Insert Java Functions:	526
Manually Configure Script Generation Settings	527
Compiling and Running a Script as Part of a Package	528
Java Icon Reference List	529
Java Custom Filters Overview	529
Java Custom Filters - Determining which Elements to Include	530
Create a Custom Java Filter	531
Define a Custom Hook File	531
Hook File Structure	532
Troubleshooting and Limitations - Java Record Replay and Java Vuser	535
Specifying connection timeouts	535
Java Vuser Protocol	536
Manually Programming Java Scripts - Overview	536
Java Protocol Programming Tips	537
Running Java Vuser Scripts	538
Opening Java Vuser Scripts in Eclipse	539
Compiling and Running a Script as Part of a Package	540
Manually Create a Java Script	540
Enhance a Java Script	543
Troubleshooting and Limitations - Java Record Replay and Java Vuser	547
Specifying connection timeouts	547
Java over HTTP Protocol	548
Java over HTTP Protocol Overview	548
Viewing Responses and Requests in XML Format	549
Record with Java over HTTP	549
Correlating Java over HTTP	551
Debug Java over HTTP scripts	552

Insert Parameters into Java over HTTP Scripts	553
Troubleshooting and Limitations for Java over HTTP	553
LDAP Protocol	555
LDAP Protocol Overview	555
LDAP Protocol Example Script	555
Defining Distinguished Name Entries	556
LDAP Connection Options	557
Troubleshooting and Limitations - LDAP	558
Mailing Service Protocols	559
Mailing Service Protocols Overview	559
IMAP Protocol Overview	559
MAPI Protocol Overview	560
POP3 Protocol Overview	561
SMTP Protocol Overview	561
Message Protocols	562
MMS (Multimedia Messaging Service) Protocol Overview	562
Run an MMS Scenario in the Controller	563
Mobile Protocols	563
Select a Recording Method for Mobile Applications	563
Web HTTP/HTML	563
TruClient - Native Mobile	564
TruClient - Mobile Web	564
SMP (SAP Mobile Platform)	564
Speed Simulation for Mobile Vuser Scripts	565
Web - HTTP/HTML - Recording Methods for Mobile Applications	566
TruClient - Mobile Web Protocol	567
TruClient - Mobile Web Protocol Overview	567
How to Record a Script with TruClient - Mobile Web	567
Add, Remove, and Import Mobile Device Settings for TruClient - Mobile Web	568
Create a Custom Device Using the Mobile Device Manager	568
Remove a Mobile Device	568
How to Import a Mobile Device Settings to Your Script	568
Mobile Device Dialog Box	568
SMP (SAP Mobile Platform) Protocol	569
MQTT Protocol	570
Create an MQTT script	570
Working with multi-protocol scripts	571
Ready-to-use MQTT script template	572
.NET Protocol	574
.NET Protocol Overview	574
Considerations for Working with the .NET Protocol	574
Viewing Data Sets and Grids	575
Recording WCF Duplex Communication	576

Replacement of the Callback in the Script	578
Asynchronous Calls	580
Recording Dual HTTP Bindings	581
Connection Pooling	581
Debugging .NET Vuser Scripts	582
.NET Filters Overview	583
Guidelines for Setting .NET Filters	584
.NET Filters - Advanced	587
Reference List Dialog Box [.NET Protocol]	588
Add Reference Dialog Box [.NET Protocol]	589
Create a New Filter Dialog Box [.NET Protocol]	589
Configure Application Security and Permissions	590
Troubleshooting and Limitations - .NET	591
Replay Limitations	591
Recording Limitations	591
Script Editing Limitations	593
Oracle NCA Protocol	593
Oracle NCA Protocol Overview	593
Oracle NCA Protocol Example Scripts	594
Oracle NCA Record and Replay Tips	594
Pragma Mode	595
Enable the Recording of Objects by Name	596
Launch Oracle Applications via the Personal Home Page	598
Oracle - Troubleshooting and Limitations	599
RDP Protocol	602
RDP Protocol - Overview	602
RDP Recording Tips	602
Single vs. Multi-Protocol Scripts	602
Record into Appropriate Sections	603
FIPS	603
Run a Clean Session	603
Explicit Mouse Clicks	603
Using Windows Logo key combinations	603
Synchronizing using Windows 8 apps	604
Working with Clipboard Data (RDP Protocol)	604
Correlating Clipboard Parameters	606
RDP Snapshots - Overview	606
Image Synchronization Overview (RDP)	607
Image Synchronization Tips (RDP Protocol)	607
Image Synchronization - Shifted Coordinates (RDP Protocol)	608
Setting Security Levels in RDP Vuser Scripts	609
RDP Agent (for Microsoft Terminal Server) Overview	612
Tips for Using the Agent for Microsoft Terminal Server	612

Enhancements to RDP functionality	613
Install/Uninstall the RDP Agent	614
Install the RDP Agent (Agent for Microsoft Terminal Server)	614
Use the RDP Agent (Agent for Microsoft Terminal Server)	614
Uninstall the RDP Agent (Agent for Microsoft Terminal Server)	615
Add Image Synchronization Points to a Script	615
Failed Image Synchronization Dialog Box (RDP Protocol)	615
Troubleshooting and Limitations for RDP	616
RTE Protocol	618
RTE Protocol Overview	618
Working with Ericom Terminal Emulation	619
SSL and SSH Support for Ericom	620
Typing Input into a Terminal Emulator	620
Setting the Timeout Value for TE_type	621
Allowing a Vuser to Type Ahead	622
Generating Unique Device Names	622
Setting the Field Demarcation Characters	623
Reading Text from the Terminal Screen	624
RTE Synchronization Overview	625
Synchronizing Block-Mode (IBM) Terminals	626
Synchronizing Character-Mode (VT) Terminals	628
Wait for the cursor to appear at a specific location	628
Instruct VuGen to automatically generate and insert TE_wait_cursor statements while recording ...	629
Wait for text to appear on the screen	629
Instruct VuGen to automatically generate and insert TE_wait_text statements while recording	630
Wait for the terminal to be silent	631
Map Terminal Keys to PC Keyboard Keys	631
Record RTE Vuser Scripts	632
Implement Continue on Error	634
Troubleshooting and Limitations - RTE	634
IP Spoofing	635
Disconnection Failures	635
Initialization Failures	635
SAP Protocols	635
Selecting an SAP Protocol Type	635
SAP GUI Protocol	636
SAP Web Protocol	639
Replaying SAP GUI Optional Windows	640
Configure the SAP Environment	640
Use IPV6 with a SAP installation	641
Check that SAPGUI Scripting API is enabled	641
Check the SAP GUI for Windows Client Patch Level	641
Check the Patch Level	641

Check the Kernel Patch Level	641
Check the R/3 Support Packages	642
Enable scripting on the SAP Application Server	643
Enable scripting on SAP GUI 6.20 client	645
Examine the hierarchy of GUI Scripting objects - Optional	646
Record SAP GUI Scripts	646
Replay SAP GUI Scripts	648
Run SAP GUI Scripts from the Controller	649
Enhance SAP GUI Scripts	650
Additional SAP Resources	654
Troubleshooting and Limitations for SAP	654
General SAPGUI limitations	654
I was able to record a script, but why does replay fail?	655
Why were certain SAP GUI controls not recorded?	655
Why can't I record or replay any scripts in VuGen?	655
What is the meaning of the error popup messages that are issued when I try to run the script?	655
Can I use the single sign-on mechanism when running a script on a remote machine?	656
Can VuGen record all SAP objects?	656
Are all business processes supported?	656
When I go to the Auto Logon node of the Recording Options, why is the list of server names empty?	656
Siebel Web Protocol	656
Siebel Web Protocol Overview	656
Siebel Web Recording Options and Runtime Settings	657
Record Transaction Breakdown Information	657
Siebel Web - Troubleshooting and Limitations	658
Silverlight Protocol	660
Silverlight Protocol - Overview	660
Silverlight - Troubleshooting and Limitations	660
Teradici PCoIP (PC over IP) Protocol	661
PCoIP client	661
Record a PCoIP test	661
Enhance the recorded script	662
Sample Script	662
TruClient Protocol	663
Convert TruClient scripts to TruClient - Web	663
Converting multiple scripts	663
Converting a single script	663
Web - HTTP/HTML Protocol	664
Web - HTTP/HTML Protocol - Overview	664
When should you use the Web - HTTP/HTML Vuser protocol?	664
Web - HTTP/HTML Vuser Technology	664
Learn how to develop a Web - HTTP/HTML Vuser script	664

JavaScript Web HTTP Scripts	666
Recording your Vuser Script in JavaScript	667
Converting a C Script to JavaScript	667
JavaScript Function Libraries	668
Using the VuGen JavaScript Engine	669
JavaScript Engine: XMLHttpRequest Example	674
Convert a Web - HTTP/HTML Vuser Script into a Java Vuser Script	675
Create a Script for a REST API	676
REST API components	676
Using the REST Step Properties dialog box	676
Create script steps for a Web - HTTP/HTML script that calls a REST API	678
Using Emulation to Record Mobile Applications	679
Recording with a Google Android Emulator	680
Record HTTP/2	683
Record and Replay on servers with SNI Enabled	684
Additional SNI Guidelines	684
Create a Vuser Script by Analyzing a Captured Traffic File	685
Create Web - HTTP/HTML scripts from TruClient Scripts	686
Record Streaming Media in Web - HTTP/HTML	686
Record Applications Using Smooth Streaming	687
Common Web Recording Problems	688
No events are being recorded	688
Very few events are being recorded	689
Specific events are not being recorded	689
Application hangs during recording	689
Wrong Server Certificate	690
Browser crashes during Ajax Click and Script Recording	690
Troubleshooting and Limitations - Web - HTTP/HTML Protocol	690
Web Protocols (Generic)	691
Web Protocols - Overview	692
Web Vuser Technology	692
Web Vuser Types	693
Text and Image Verification (Web Vuser Scripts) - Overview	694
Understanding Web Text Check Functions	695
web_reg_find	695
web_find	695
web_global_verification	695
Add Text Checks and Image Checks (Web Vuser Protocols)	696
Add a Text Check While Recording	696
Add a Text Check After Recording	696
Add Other Text Checks After Recording	696
Add an Image Check After Recording	696
Web Snapshots - Overview	697

Browser Emulation - Overview	698
What is a User-Agent String?	699
Emulating a specific browser	699
What information is included in Web Vuser's user-agent string?	700
Maximum number of concurrent connections	700
Perform Load Testing with nCipher HSM	701
Working with Cache Data	702
Create the cache file	702
Load the cache file into a Vuser	703
Insert Caching Functions	704
Data Format Extensions (DFEs) - Overview	704
When to use DFEs	704
DFE support	705
DFE chains	706
Replaying Vuser scripts that contain DFEs	707
Implement Data Format Extension (DFE) Support	707
Define a Chain of DFEs	708
Adding a DFE chain	708
Adding DFEs to the new DFE chain	709
Enable DFE Support	709
Configure DFE Support	710
Apply DFE Chains to Sections of the HTTP Message	711
How DFEs Modify a Vuser Script	712
Data Format Extension List	713
Applying DFEs to a String	714
Google Web Toolkit - Data Format Extension (GWT-DFE) - Overview	715
Implementing GWT-DFE Support	722
Prerequisites for implementing GWT-DFE support	722
Recording GWT-DFE Headers	723
Applying GWT-DFE chains	723
Troubleshooting - Data Format Extension (DFE)	724
Web Services Protocol	725
Web Services - Adding Script Content	725
Web Service Testing Overview	725
Adding Web Service Script Content - Overview	725
Recording a Web Services Script	725
Adding New Web Service Calls	726
Importing SOAP Requests	726
Analyzing Server Traffic	727
Script Integration	727
Web Service Call Attachments	727
Special Argument Types	728
Server Traffic Scripts Overview	731

Filtering Traffic	732
Data on Secure Servers	733
Add Content	733
Assign Values to XML Elements	735
Generate a Test Automatically	736
Create a Script by Analyzing Traffic (Web Services)	737
Specify Services Screen	737
Specify Application to Record Dialog Box	738
Import SOAP Dialog Box	739
New Web Service Call Dialog Box	740
Add Input Attachment Dialog Box	747
Add Array Elements Dialog Box	748
Process Base64 Data - Simple Data Dialog Box	748
Process Base64 Data - Complex Data Dialog Box	749
Aspects List	750
Specify Services Screen	751
Specify Traffic Information Screen	752
SSL Configuration Dialog Box	752
Web Services - Preparing Scripts for Replay	753
Preparing for Replay Overview	753
Testing Web Service Transport Layers Overview	754
Sending Messages over HTTP/HTTPS	754
JMS Transport Overview	754
JMS Script Functions	755
JMS Message Structure	756
Asynchronous Messages Overview	756
Sending Asynchronous Calls with HTTP/HTTPS	756
WS-Addressing	757
Customizing Overview	758
User Handlers	758
Custom Configuration Files	760
User Handler Examples	760
Prepare Scripts for Replay	763
Send Messages over JMS	764
Send Messages over HTTP/S	765
Define a Testing Method	766
Add a Database Connection	768
Create a User Handler	768
Customize Configuration Files	771
Web Services Snapshots - Overview	772
Database Connection Dialog Box	774
Connection String Generator Dialog Box	774
Web Services - Managing Services	775

Managing Services Overview	775
Description tab	776
Operations tab	777
Connection Settings tab	777
UDDI Data tab	777
Protocol and Security Settings tab	777
Importing Services	778
Comparison Reports	778
Web Reference Analyzer	779
Add and Manage Services	779
Analyze WSDL Dependencies	781
Manage Services Dialog Box	781
Connection Settings Dialog Box	784
Import Service Dialog Box	784
Search for Service in UDDI Dialog Box	785
XML/WSDL Comparison Dialog Box	786
WSDL Reference Analyzer Dialog Box	786
Web Services - Security	787
Setting Security Overview	787
Security Tokens and Encryption	788
Available Security Tokens	788
Adding the Security Policy	789
Working with Message Signatures and Encrypted Data	790
SAML Security Options	790
Security Scenarios Overview	791
Choosing a Security Model	791
Private, Imported, and Shared Scenarios	792
Scenario Categories	792
WCF Scenario Settings	794
The WsHttpBinding Scenario	795
The Federation Scenario	796
The Custom Binding Scenarios	797
WCF Extensibility	798
Binding	798
Channel	798
Behavior	798
Examples of Channel and Behavior	798
Preparing Security Scenarios for Running	799
Parameterizing Security Elements	799
Protecting Custom Headers	799
Emulating Users with Iterations	800
Add Security to a Web Service Script	801
Customize the Security	801

Add SAML Security	804
Create and Manage Security Scenarios	804
Parameterize Security Elements	806
Set Security Properties Dialog Box	807
WS—Security Tab	807
WS Addressing	810
Security Scenario Editor Dialog Box	810
Advanced Settings Dialog Box	811
Select Certificate Dialog Box	815
Web Services Security Examples	816
Troubleshooting and Limitations for Web Services	818
Windows Sockets Protocol	820
Recording Windows Sockets - Overview	820
Translation Tables	820
Windows Sockets Data	821
Windows Sockets Snapshots - Overview	822
Displaying Buffer Data	822
Buffer Views	822
Status Bar	823
Navigating within the Buffer Data	823
Creating Correlations from the Buffer Data	824
Data Navigation Tools	824
Buffer Data Editing	824
Record a Windows Sockets Script	824
View and Modify Windows Sockets Buffers	826
Modifying buffer data	826
View and modify the data in the data.ws file	826
View the data in the Snapshot pane	826
Navigate within the snapshot data	826
Insert data into a buffer	827
Copy and paste blocks of data	827
Data Buffers	828
Go To Offset Dialog Box	829
Customize Your Scripts	831
Create a PCAP File	831
Create a capture file	831
Troubleshooting missing packets	832
Manually Program a Script using the VuGen Editor	833
Manually Programming Scripts - Overview	833
Programming Vuser Actions	834
Create a Template	835
Define Transaction and Insert Rendezvous Points Manually	835
C Vuser Scripts	836

Java Vusers	837
.NET Vusers	837
Troubleshooting and Limitations - Programming	838
Framework 4.5 for .NET scripts	838
Framework 3.5 for .NET scripts	838
Missing references for scripts created in Visual Studio	838
Create Scripts in External IDEs	839
Creating Vuser Scripts or Unit Tests in Visual Studio or Eclipse	839
Create a Vuser Script in Visual Studio	840
Create a Vuser Script in Eclipse	841
Develop a Unit Test Using Visual Studio (NUnit test)	842
Develop a Unit Test Using Eclipse (JUnit or Selenium test)	843
Use DLLs and Customize VuGen	844
Calling Functions from External DLLs	844
Load a DLL Locally	844
Load a DLL Globally	845
VuGen File and Library Locations	846
Storing Runtime Settings in External Files	847
Command Line Parameters	847
Create and Run Scripts in Linux	848
Creating and Running Scripts in Linux - Overview	848
Compile Scripts Manually on Linux	848
Run a Vuser Script from a Linux Command Line	849
Program with the XML API	851
Programming with the XML API Overview	851
Using XML Functions	851
Reading Information from an XML Tree	851
Writing to an XML Tree	852
Specifying XML Function Parameters	853
Defining the XML Element	853
Querying an XML Tree	854
Multiple Query Matching	854
XML Attributes	855
Structuring XML Scripts	855
Enhancing a Recorded Session with XML	856
Use Result Parameters	859
Non-English Language Support	861
Non-English Language Support Overview	861
Page Request Header Language	862
Convert Encoding Format of a String	862
Convert Encoding Format of Parameter Files	863
Record Web Pages with Foreign Languages	864
Automatically Record Foreign Language Web Pages.	864

Manually Record Foreign Language Web Pages	865
Troubleshooting and Limitations for Non-English Languages	865
Share Software Content Resources	868
Troubleshooting and Limitations for VuGen	868
Internet Explorer and Windows Server Machines	868
Error Messages	868
Slow replay of JavaScript language scripts	869
Installing and Upgrading JVMs	869
McAfee Compatibility Issues	869
Additional Components	869
Where to install additional components	870
Standalone Applications	874
Protocol SDK	875
Install the Protocol Library Package	875
Upgrade Protocol SDK Projects from Visual Studio 2012	876
Installing the Virtual Table Server (VTS)	876
Installing the Microsoft Terminal Server Agent	877
Install the Agent for Microsoft Terminal Server	877
Uninstall the Agent for Microsoft Terminal Server	877
Troubleshooting and Limitations for Additional Components	877
Function Reference	879
Send Us Feedback	880

Welcome to the VuGen User Guide

Welcome to HPE Virtual User Generator, VuGen, HPE's tool for creating Vuser scripts. You use VuGen to develop a Vuser script by recording a user performing typical business processes. The scripts let you emulate real-life situations.

You use the scripts created with VuGen in conjunction with other products — HPE LoadRunner, HPE Performance Center, and HPE Business Service Management.

HP LoadRunner, a tool for performance testing, stresses your entire application to isolate and identify potential client, network, and server bottlenecks.

HP Performance Center implements the capabilities of LoadRunner on an enterprise level.

HP Business Service Management helps you optimize the management and availability of business applications and systems in production.

You can access various additional documentation for LoadRunner from **Start > All Programs > HPE Software > HPE LoadRunner > Documentation**. In icon-based desktops such as Windows 8, search for the User Guide.

What's New

Check out below the many features that are introduced or improved in LoadRunner 12.55.



Note: For more details about supported integrations and technologies, see the [System Requirements](#) (previously named Product Availability Matrix or PAM).

New - MQTT Protocol

- **MQTT** is a lightweight messaging protocol, primarily designed for machine-to-machine (M2M) or Internet of Things (IoT) connectivity. With LoadRunner's MQTT scripting protocol, you can create Vuser scripts to run performance tests for MQTT communications.

See "[MQTT Protocol](#)" on [page 570](#).

New - support for JMeter tests

- The new **JMeter tests** integration (provided as beta version) enables you to run JMeter scripts from inside Controller, alongside Vuser scripts, and view JMeter test results in addition to LoadRunner measurements. Currently, JMeter test execution is completely **free** and doesn't require any additional license.

See JMeter Tests.

New supported technologies and platforms

- Improved **Java 8** support.
- Support for new **operating systems**:
 - Windows 10 (Creators)
 - Windows Server 2016
 - Ubuntu 16.04
- **Eclipse**:
 - Support for **64-bit** version.
 - Support for **Neon.2** version. See ["Develop a Unit Test Using Eclipse \(JUnit or Selenium test\)" on page 843](#).
- **Visual Studio**:
 - Support for **64-bit** version.
 - Support for NUnit 3.2 **Test Adapter**. See ["Develop a Unit Test Using Visual Studio \(NUnit test\)" on page 842](#).
- Support for TruClient on **Chromium 55**.

Protocol enhancements

- New **MQTT** protocol, described above in ["New - MQTT Protocol" on the previous page](#).
- **Web HTTP/HTML** protocol:
 - Support for **HTTP/2** 64-bit record and replay.
 - Support for **HTTP/2** replay on Linux.
 - **HTTP/2** stability and reliability improvements.
 - **REST API** editor is now available in all Web-based protocols, including Web – HTTP/HTML, Web Services, Oracle NCA, Oracle - Web, Siebel - Web, SAP - Web, Flex.

See ["Record HTTP/2 " on page 683](#) and ["Create a Script for a REST API" on page 676](#).
- **Web-based protocols usability** enhancements:
 - **Proxy recording**:
 - Support for **NTLM** authentication.
 - Ability to select the **network interface card** (NIC).

See ["Recording via a Proxy - Overview" on page 214](#).
 - **Correlations**:
 - The **correlation rules** are updated, and include new correlation rules. See ["Correlations > Rules Recording Options" on page 151](#).
 - VuGen includes a new **correlation API**, the attribute-based correlation (ABC) API, enabling dynamic value extraction from HTML documents. See ["Design Studio \[Correlation Tab\] Dialog Box" on page 242](#).

- Now includes the ability to search and apply correlations in **HTTP headers**.
- Support for web_reg_save_param_ex in **Flex** protocol.
- **HTTP video streaming** enhancements. See ["Record Streaming Media in Web - HTTP/HTML" on page 686](#).
- The **Cipher list** is enhanced, to display only the ciphers which are relevant for the selected SSL level, plus the addition of new ciphers in the list, and customization support for the recording option. See ["Server Entry - Port Mapping Dialog Box" on page 187](#).
- **Google Web Toolkit** (GWT):
 - Support for GWT **Request Factory**.
 - Support for GWT **2.8.0**.See ["Google Web Toolkit - Data Format Extension \(GWT-DFE\) - Overview" on page 715](#).
- **HAR files** can now be generated from the Replay Summary pane (for Web HTTP/HTML). See ["Replay Summary Pane" on page 108](#).
- **PCAP traffic analysis** now supports Wireshark 2.2.x (up to and including version 2.2.7). See ["Create a PCAP File" on page 831](#).
- Support for **Dynatrace** application performance monitoring, with new runtime settings enabling you to generate the x-dynaTrace header for Dynatrace monitoring. See Preferences View - Internet Protocol, ["Advanced" on page 292](#) section.
- When importing a **saz (Fiddler)** file, the import now includes comments that are converted and included in the generated script. See ["Create a Vuser Script by Analyzing a Captured Traffic File" on page 685](#).
- **Web Services** protocol:
 - Support for **MTOM attachments**.
 - Protocol **security** improvements, including **JKS** support.See ["Web Services Protocol" on page 725](#).
- **Teradaci PCoIP** protocol:
 - Support for **VMware Horizon View**.
 - **Network Virtualization** support in Per Load Generator mode.
 - Protocol **API** enhancements.See ["Teradaci PCoIP \(PC over IP\) Protocol" on page 661](#).
- **Java Record Replay** protocol:
 - Support for record and replay with **Java 8** for **JDBC** applications.
 - Support for record and replay with **Java 8** (in addition to Java 7) for **JMS** applications.
 - Support for recording of multi-threaded applications.See ["Java Record Replay Protocol Overview" on page 516](#).
- **Java Vuser** protocol:
 - Support for **64-bit** recording.
 - Protocol **API** enhancements.

See ["Java Vuser Protocol" on page 536](#) and the Function Reference (**Help > Function Reference**).

- The **Java over HTTP** protocol now supports IP spoofing. See ["Java over HTTP Protocol" on page 548](#).
- The **Citrix** protocol includes API enhancements. See [Citrix](#) in the Function Reference.
- The **SAP GUI** protocol includes usability improvements. See ["SAP Protocols" on page 635](#).
- The **FTP** protocol now has improved file upload and MLSD command support.
- The **ODBC** protocol now supports 64-bit record and replay. See ["Database Protocols" on page 478](#).

TruClient enhancements

For a full list of TruClient updates, see the [TruClient What's New](#).

- You can now convert **TruClient actions to code** (provided as tech preview version). This enables you to take advantage of the fast script development and processing available with TruClient when you create a script. You can then convert selected actions to code, making it easier to add complex logic and customized functions to the script. See [Convert TruClient Actions to Code](#).
- **TruClient Lite** now includes the ability to import client-side certificate files using the Download Certificate Generator tool. See [The Scripts Management Dialog](#).

TruClient Lite is an extension to the Chrome browser that uses your local installation of Chrome to develop scripts. Find it in the [Chrome Web Store](#).
- The new AUT (Application Under Test) API enables common access to the **AUT.window** and **AUT.document** objects. See [Using JavaScript in the AUT Window](#).
- The new **Play Until This Step** option enables script replay to stop before a selected step. See [Context Menu](#).
- The enhanced **TruClient Native Mobile** protocol now includes support for hybrid and multiple applications, new steps and General Settings, and more. See the [Mobile Center Help](#).
- **TruClient Standalone** includes various usability improvements and fixes. TruClient Standalone enables you to create scripts independent of a VuGen installation. For details, see [TruClient Standalone Launcher](#).
- Improved ability to record **SAPUI5** applications, so that recording is now supported with all browsers and with greater usability.
- TruClient includes multiple **record and replay** enhancements, improving object recognition and handling, and making the record and replay process more robust and efficient.
- Improved **GUI**, providing better performance, especially when working with large scripts.

VuGen enhancements

- The new **Recording Summary Report** provides a single-view summary of the recorded data for Web HTTP/HTML scripts, and enables you to manage traffic filtering. The report displays general information about your recording session, and enables filtering and drilldown for hosts, headers, content types, and other recording information. See ["Recording Summary Report" on page 137](#).

- The new **Check for Update** feature automatically provides information on new versions and product updates, at VuGen startup. See ["Introducing VuGen" on page 34](#).
- **Multiple users** can now log into the same machine using Remote Desktop, open their own instance of VuGen, and start browser recording.
- The **Recording Options** and **Runtime Settings** include various enhancements and fixes to improve the user experience.
- VuGen includes many other scripting productivity improvements, as well as general performance and usability enhancements.

Controller and Analysis enhancements

- New **JMeter tests** integration, described above in ["New - support for JMeter tests" on page 28](#).
- The new **Dynatrace** integration provides information on monitored applications, using Dynatrace measurements provided by the Dynatrace server. See Dynatrace Monitor.
- You can now view a **New Relic** graph in Analysis, displaying metrics from the New Relic server. See New Relic Graph.
- **HTTP video streaming** monitors and graphs are now available, enabling the monitoring and analysis of data for streaming media. See HTTP Streaming Monitoring.
- The new **MQTT monitors and graphs** enable the monitoring and analysis of MQTT message flow and throughput between the MQTT broker and the client during the load test run. See MQTT Statistics Monitoring.

Network Virtualization enhancements

- **Network Virtualization:**
 - Support for Network Virtualization emulation on **Linux** load generators.
 - Includes **Latency compensation**, enabling compensation for extraneous network latency.
See
 - Network Virtualization Integration
 - [Network Virtualization for LoadRunner & Performance Center Help](#)
- **NV Insights Report** (previously NV Analytics Report):
 - Added support for the **HTTP/2** protocol.
 - Now includes **Client-side breakdown** data for **TruClient** scripts (added to the HTTP Waterfall report), providing links between client activities and network traffic.
 - Displays **TruClient** steps, events, pages, and snapshots.
 - New **Resource Analysis** page to help identify issues related to the largest resources, and the resources with the longest download time.
 - Now includes **Transaction status**, and errors that occurred during the transaction run.

See:

- ["NV Insights Report" on page 312](#) for VuGen
- NV Insights Report for Controller
- Information on the NV Insights Report in [Network Virtualization for LoadRunner & Performance Center Help](#)

Integration enhancements

For details about supported HPE product versions for integration with LoadRunner, see the [Integration Support Matrices](#).

- **Jenkins** integration:
 - Support for Jenkins 5.2, including the **Build Pipeline** plug-in.
 - **Correlation** performance improvements.

See [HPE Application Automation Tools](#).

- The new [ADM Marketplace](#) has replaced the HPLN integration. In the ADM Marketplace you can find product add-ons and plugins, code samples, extra protocols, and additional downloads for your HPE Performance Engineering applications and components.

The Marketplace content is created by the HPE team and by community members. You are invited to contribute your own content and experience to the huge Performance Engineering community.

VuGen

The Virtual User Generator (VuGen) enables you to record and develop scripts for load testing with HPE products.

To learn more about VuGen, see ["Introducing VuGen" below](#).

Introducing VuGen

Virtual User Generator, or **VuGen**, is a primary tool for creating testing scripts that emulate behavior of real users on your system.

Overview

When testing or monitoring an environment, you need to emulate the true behavior of users on your system. HPE testing tools emulate an environment in which users concurrently work on, or access your system.

To perform this emulation, the human is replaced with a virtual user, or a *Vuser*. The actions that a Vuser performs are typically recorded in a *Vuser script*.

You use VuGen to develop a Vuser script by recording a user performing typical business processes. The Vuser scripts let you emulate real-life situations.

Use VuGen scripts to test your environment

You use the scripts created with VuGen in conjunction with other HPE products, as follows:

- **LoadRunner**, a tool for performance testing, stresses your entire application to isolate and identify potential client, network, and server bottlenecks.
- **StormRunner Load** is a cloud-based load testing solution that allows Agile development teams to create effective tests. Get a free trial here: [StormRunner Load Home](#).
- **Performance Center** implements the capabilities of LoadRunner on an enterprise level.
- **Application Performance Management (APM; formally Business Service Management)** helps you optimize the management and availability of business applications and systems in production. VuGen is used in conjunction with the following APM components:
 - **Business Process Monitor (BPM)** software is a synthetic monitoring solution that simulates business transactions—whether or not real users are active. You use VuGen to create scripts for BPM, in order to reuse assets in testing and production environments.
 - **Real User Monitoring (RUM)** software monitors application performance and availability on business critical application services, for all users. You use VuGen to capture and replay user sessions, and to create test scripts that reflect real user behavior.
- **AppPulse** enables you to monitor applications across traditional, mobile, virtualized, and cloud environments. You can use VuGen to record scripts for AppPulse, across a range of protocols. The scripts are imported into AppPulse and used for availability and performance monitoring, by automatically reproducing the activity of the user.

VuGen Updates

To check for VuGen updates, use the **Check for Updates** feature available from VuGen's **Help** menu.

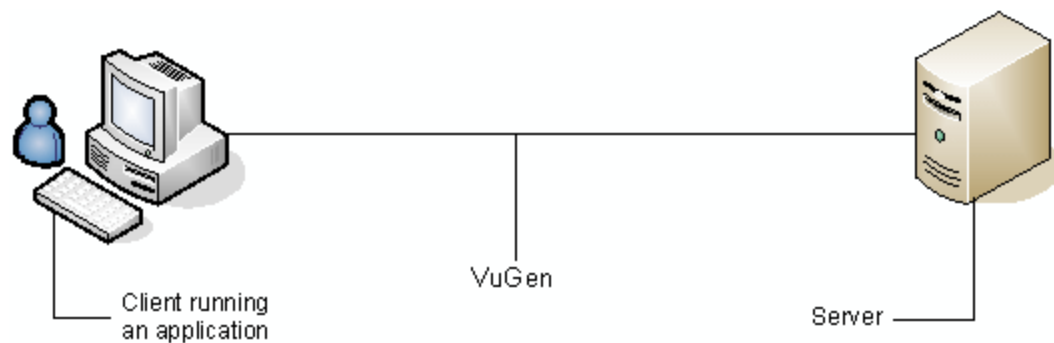
Alternatively, you can configure VuGen to notify you about updates automatically on startup. For details, see [Update Messages](#).

Vusers

Vuser Technology

You use VuGen to develop a Vuser script by recording a user performing typical business processes on a client application. VuGen records the actions that you perform during the recording session, recording only the activity between the client and the server.

During recording, VuGen monitors the client and traces all the requests sent to and received from the server.



After the recording, VuGen generates various functions that define the actions performed during the recording session. VuGen inserts these functions into the VuGen editor to create a basic Vuser script.

Instead of having to manually program the application's API function calls to the server, VuGen automatically generates functions that model and emulate real world situations.

VuGen not only records Vuser scripts, but also replays them. Replaying scripts from VuGen is useful for debugging. It enables you to determine how a Vuser script will run when it is executed as part of a larger test.

During playback, Vuser scripts communicate directly with the server by executing calls to the server API functions. When a Vuser communicates directly with a server, system resources are not required for the client interface. This lets you run a large number of Vusers simultaneously on a single workstation, and enables you to use only a few testing machines to emulate large server loads.



In addition, since Vuser scripts do not rely on client software, you can use Vusers to check server performance even before the user interface of the client software has been fully developed.

You can use the Vuser scripts in several HPE products that incorporate scripts.

One option is to add them to a LoadRunner scenario using the Controller. While running the Vusers from the Controller, you gather information about the system's response. After the test run, you can view this information with the Analysis tool. For example, you can observe how a server behaves when one hundred Vusers simultaneously withdraw cash from a bank's ATM.

VuGen only records scripts on Windows platforms. However, a recorded Vuser script can be replayed on both Windows and Linux platforms.

You can also program Vuser scripts in your native programming application, such as MS Visual Studio. To access the API, install the appropriate IDE add-in provided on the product's DVD.

Vuser Types

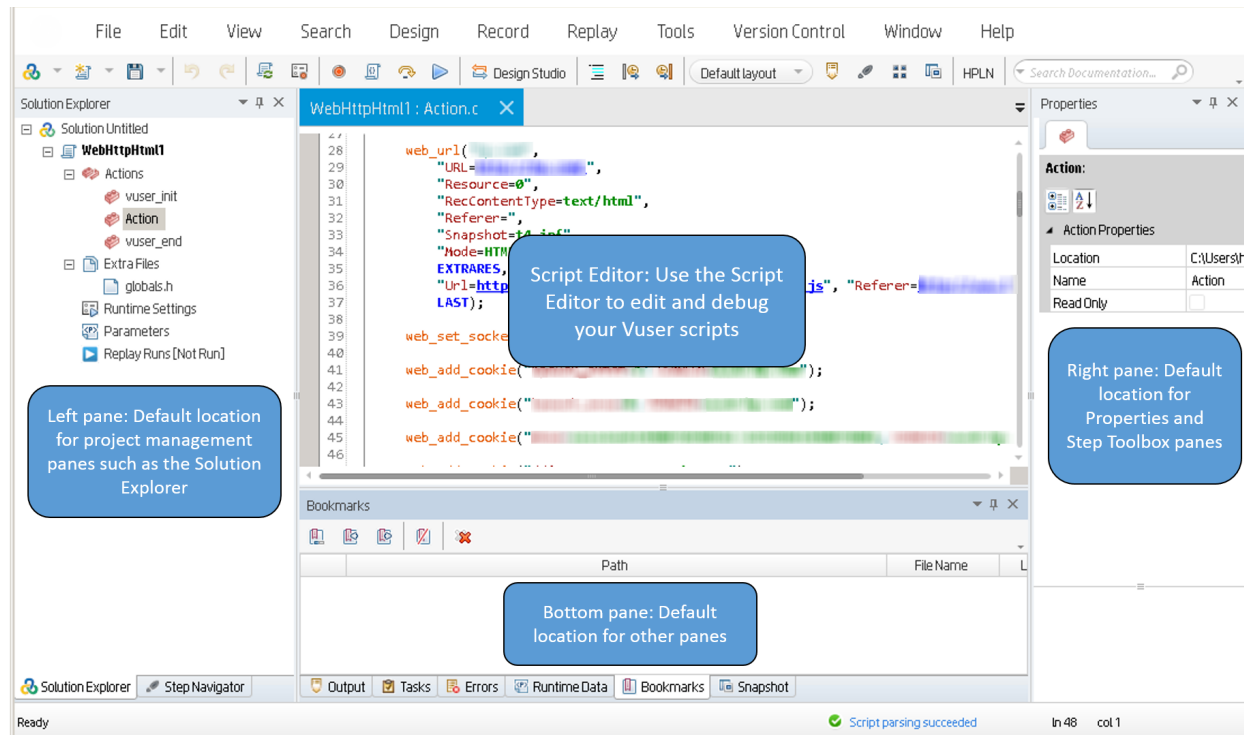
Vuser Type	Description
Protocol Based Vusers	<p>Vusers that run scripts developed in VuGen. VuGen supports most of the common communication protocols. For a complete list of the available Vuser protocols, see "VuGen Protocols" on page 417.</p> <p>You can create a Vuser script using a single protocol or multiple ones.</p>
Unit Test Based Vusers	<p>Vusers running unit tests in the form of .dll or .jar/.class files, created in supported versions of Microsoft Visual Studio or Eclipse.</p> <p>To create these tests, install the IDE for Developer add-in, available in the Additional Components folder of the product's DVD.</p>
GUI Vusers	<p>Vusers that perform functional testing on a user interface level. You create these tests using HPE Functional Testing software, such as UFT (Unified Functional Testing).</p> <p>You can only run a single GUI Vuser on a Windows-based load generator. To run multiple GUI Vusers you can use Citrix sessions.</p>

VuGen User Interface

The VuGen editor and panes are the environment you will be working in while you record, replay, and debug a Vuser script.

VuGen Workspace

The VuGen workspace enables you to record, edit and debug your Vuser script. The VuGen workspace is divided up as follows:



Script Editor

VuGen's Editor enables you to edit recorded scripts and other supplementary files such as header files. You can open multiple files simultaneously, navigating tab by tab. The editor also supports multiple programming languages, code coloring, code folding (enables you to selectively hide and display sections of your code), code completion and tooltips for C scripts. For details see ["Editor Pane" on page 54](#).

Project Management Panes

The project management panes include the Solution Explorer, the Step Navigator and the Outline pane. By default they all appear on the left side of the workspace. The Solution Explorer enables you to easily organize and navigate through script entities, enhancing the recording, replay and debug process. You can create a solution containing multiple scripts of different protocols related to a full-cycle business process. Each script entity includes extra files (such as header files), runtime settings, parameters, and replay runs. For details see ["Solution Explorer Pane" on page 46](#).

The Step Navigator enables you to navigate to a selected step in your script. If your script contains many steps, you can use the search box to search for matching text in the different parts of the steps. For details see ["Step Navigator Pane" on page 52](#).

Window Panes

VuGen has a number of window panes which by default are displayed at the bottom of the workspace. Each window pane deals with one specific aspect of working with the script. For example, the Errors pane displays all errors in the script.

The following table describes each pane and provides a short use case scenario.

Pane	Used For:	For Details:
Bookmarks	Specifying a location in a script so that you can easily find it later on for editing.	See "Bookmarks Pane" on page 61
Errors	Displaying script errors, warnings and messages generated from script replay. Creating custom filters for error messages.	See "Errors Pane" on page 72
Snapshot	Displays server and client data associated with a specific step in a script. The format of the data is dependent on the protocol used for creating the script.	See "Snapshot Pane" on page 62
Data Grid	Simplifying views of all recordsets associated with the script. Valid for specific protocols such as ODBC. Contains either sent or received data. Parameterizing and manipulating data displayed in the data grid.	
Tasks	Adding, editing or searching for tasks related to a script or solution.	See "Tasks Pane" on page 74
Thumbnail	Following the business process that the script has recorded.	See "Thumbnail Explorer" on page 71
Output	Event log from different operations in VuGen such as code generation, replay, and recording.	See "Output Pane" on page 77

Pane	Used For:	For Details:
Breakpoints	Managing breakpoints in Vuser scripts to help debug the scripts.	See "Breakpoints Pane" on page 78
Watch	Monitoring variables and expressions while a script runs, and is in the Paused state.	See "Watch Pane" on page 81 .
Call Stack	Viewing information about the methods and functions that are currently on the call stack of your script, or the context in which the run session was paused.	See "Call Stack Pane" on page 81

Properties Pane and Steps Toolbox

The Properties pane and Steps Toolbox are displayed on the right side of the workspace.

The Properties pane displays the selected object's properties, such as the object's location. Each object has its own specific properties list. You can sort the property list according to category or by alphabetical order.

The Steps Toolbox displays a list of API functions which you can drag and drop into your script. The API functions are divided into categories. For details on each API function see the *HPE Loadrunner API Reference Guide*. For more details on the Steps Toolbox see ["Steps Toolbox Pane" on page 60](#).

Standard Layouts

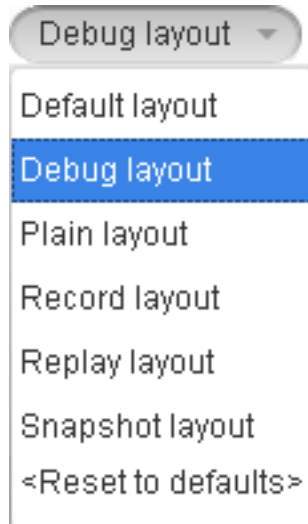
VuGen has many different window panes which you may want to display or hide based on what you are currently doing. You can also move the panes around the workspace, in order to customize the workspace layout. VuGen comes with a set of standard layouts:

- Default
- Debug
- Plain
- Record
- Replay
- Snapshot

Each layout is designed to enhance a specific phase of the Vuser script development process. For example, the Replay layout includes the panes that are most useful when you run a Vuser script: **Errors**, **Call Stack**, **Watch**, **Breakpoints**, **Output**, and **Runtime Data**.

VuGen automatically uses specific layouts during specific phases of the script development process. For example, the Record layout is used while you record a script, and the Replay layout is used when you replay a script.

The VuGen toolbar displays the layout that is currently used: **Debug layout** . To change the layout, click the **Layout** drop-down and select the required layout from the list of layouts, as shown below.



For details on how to customize VuGen layouts, see ["How to Modify the VuGen Layout" below](#).

Customizing your Workspace

You cannot add or delete a standard layout. However, you can modify most of VuGen's standard layouts to meet your specific requirements. When you modify a layout, you can add, move and resize zones, select which panes to include in each zone, and specify which of these panes is displayed by default. For task related details, see ["How to Modify the VuGen Layout" below](#). After you modify a standard layout, VuGen maintains that layout until you change the layout again or reset the default layouts.

Note: VuGen does not save any changes that you make to the Plain layout.

Restoring the layout defaults

On the VuGen toolbar, click the **Layout** drop-down, and select **Reset to Defaults**. VuGen resets all standard layouts to their default settings.

How to Modify the VuGen Layout



The VuGen window is composed of a number of zones. Each zone can contain a variety of panes, such as the Errors pane and the Snapshot pane. When more than one pane is included in a zone, the panes appear as tabs within the zone. This section describes how to customize and modify the zones and panes that appear in the VuGen window.

Move a pane to a new zone

You can move any VuGen pane to a new zone. The new zone can be either a portion of an existing zone, or it can occupy the entire left, right, top, or bottom of the VuGen window.

In the VuGen window, drag the title bar or tab of the pane that you want to move. (If the required pane is not displayed in the VuGen window, you can select it from the **View** menu.) As you drag the pane over the zones in the VuGen window, a complex marker is displayed in the center of the active zone and a simple marker appears on each edge of the VuGen window.

Note: If you drag the title bar of a zone that contains multiple tabbed panes, all of the panes in the zone move to the new zone.

Marker Type	Marker	Description
Complex marker - Current zone		Positions the selected pane in a new zone. The new zone is created in the top, bottom, left, or right of the active zone , according to the arrow marker selected when you release the mouse button.
Simple marker - VuGen window		Positions the selected pane in a new zone. The new zone is created in the top, bottom, left, or right of the VuGen window , according to the arrow marker selected when you release the mouse button.

Move a pane to an existing zone

You can move any VuGen pane from one zone to another. When more than one pane is included in a zone, the panes appear as tabs within the zone.

1. In the VuGen window, drag the title bar or tab of the pane you want to move. (If the required pane is not displayed in the VuGen window, you can select it from the View menu). As you drag the pane over the zones in the VuGen window, a complex marker is displayed in the center of the active zone.



2. Locate the cursor over the center button of the complex marker. When you release the mouse button, the selected pane is added as a tabbed pane to the selected zone.
3. Repeat this procedure for each pane you want to move.

Note: If you drag the title bar of a zone that contains multiple tabbed panes, then all the panes in the zone are moved to the selected zone.

Float and dock panes

Docked panes are fixed in a set position within the VuGen window. For example, when you move a pane to a position indicated by a marker, the pane is docked in that position.

Floating panes are displayed on top of all other windows. Floating panes can be dragged to any position on your screen, even outside the VuGen window. Floating panes have their own title bars.

- To float a pane, right-click the title bar, and click **Float**. The pane opens on top of all the other windows and panes, with its own title bar.
- To dock a pane, double-click the title bar, or right-click the title bar and select **Dock as tabbed document**. The pane returns to its previous position in the VuGen window.

Keyboard Shortcuts

This section lists the keyboard shortcuts available for the VuGen menus.

File Menu

New > Script and Solution	Ctrl+N
Open > Script/Solution	Ctrl+O
Add > New Script	Ctrl+Shift+A
Add > Existing Script	Alt+Shift+A
Close > Document	Ctrl+F4
Close > Solution	Ctrl+Shift+F4

Save Script	Ctrl+S
Save All Scripts	Ctrl+Shift+S
Reload File	Ctrl+Shift+U
Print	Ctrl+P
Exit	Alt+F4

Edit Menu

Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Del
Select All	Ctrl+A
Format > Surround with	Ctrl+J
Format > Increase Indent	Tab
Format > Decrease Indent	Shift+Tab
Folding > Toggle fold	Ctrl+Shift+M
Folding > Toggle all folds	Ctrl+Shift+L
Folding > Show definitions only	Ctrl+Shift+P
Show Function Syntax	Ctrl+Shift+Space
Complete Word	Ctrl+Space

View Menu

Solution Explorer	Ctrl+Alt+L
Search Results	Ctrl+Alt+R

Bookmarks	Ctrl+Alt+K
Steps Toolbox	Ctrl+Alt+B
Snapshot	Ctrl+Alt+P
Steps Navigator	Ctrl+Alt+S
Thumbnail Explorer	Ctrl+Alt+T
Properties	Ctrl+Alt+F4
Output	Ctrl+Alt+O
Full Screen	Alt+Shift+Return

Search Menu

Quick Find	Ctrl+F
Find Next	F3
Find Next Selected	Ctrl+F3
Find in Files	Ctrl+Shift+F
Quick Replace	Ctrl+H
Incremental Search See Incremental Search	Ctrl+E
Reverse Incremental Search See Incremental Search	Ctrl+Shift+E
Bookmarks > Toggle Bookmark	Ctrl+F2
Bookmarks > Prev Bookmark	Shift+F2
Bookmarks > Next Bookmark	F2
Go To	Ctrl+G

Design Menu

Action > Delete Action	Delete
Action > Rename Action	F2

Insert in Script > New Step	Alt+Insert
Insert in Script > Start Transaction	Ctrl+T
Insert in Script > End Transaction	Ctrl+Shift+T
Insert in Script > Comment	Ctrl+Alt+C
Parameters > Parameters List	Ctrl+L
Parameters > Create New Parameter	Ctrl+K
Parameters > Configure Parameter Delimiters	Ctrl+B
Design Studio	Ctrl+U

Record Menu

Record	Ctrl+R
Regenerate Script	Ctrl+Shift+R
Recording Options	Ctrl+F7

Replay Menu

Run	F5
Stop	Ctrl+F5
Compile	Shift+F5
Toggle Breakpoint	F9
Continue Debugging	F5
Run Step by Step	F10
Runtime Settings	F4

ALM Menu

ALM Connection	Ctrl+Q
----------------	--------

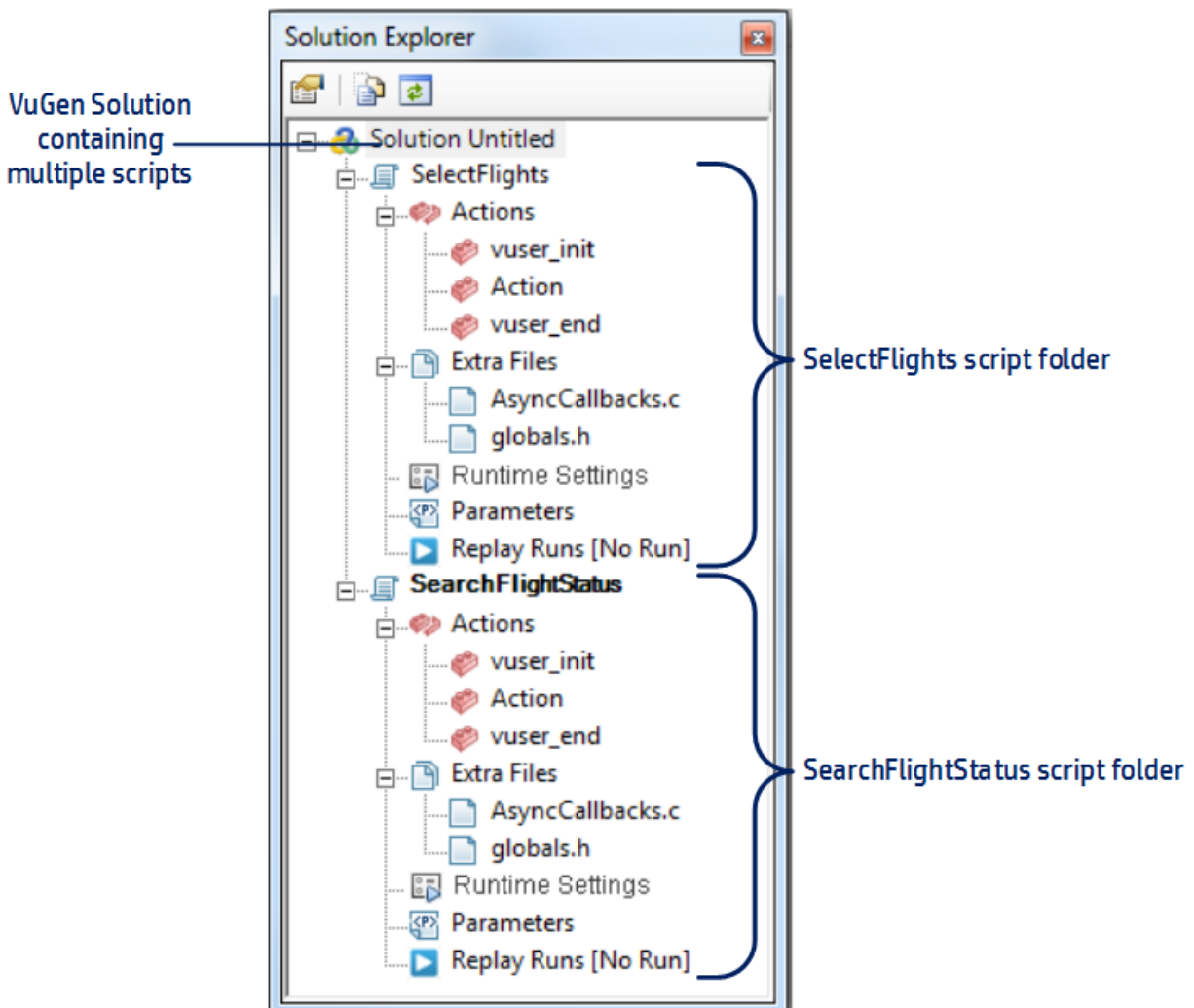
Windows Menu

Next Window	Ctrl+Tab
Prev Window	Ctrl+Shift+Tab

Solution Explorer Pane

The Solution Explorer enables you to manage your Vuser scripts. A solution contains Vuser scripts. Vuser scripts consist of script files, extra files (such as header files), runtime settings, parameters and replay runs. A solution can contain multiple scripts of different protocols.



The image below shows a Solution Explorer with two scripts.



To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> • View > Solution Explorer • Press Ctrl + Alt + L
Important information	<ul style="list-style-type: none"> • Solution Explorer is automatically displayed as part of the default layout. • You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40. • Other main interface panes such as Output, Error and Snapshot synchronize their displays based on your location in the Solution Explorer. • You can double-click an asset to activate it in the editor area or right-click to examine quick operations available for that asset. • You can bundle scripts in a solution. For example you can bundle scripts related to one business process. <div style="background-color: #e6f2e6; padding: 10px; margin-top: 10px;"> <p>Note: The solution explorer cannot be imported into any of the existing management tools such as ALM or Controller.</p> </div>
See also	<ul style="list-style-type: none"> • "VuGen User Interface" on page 37 • "Import Actions to a Script" on page 219

Understanding the Solution Explorer

Entity	Used for	Comments
Solution	Container for all script objects.	Give your solutions meaningful names, such as the name of the business process. The default solution name is "Untitled"
Scripts	Creating, editing and debugging scripts.	<p>Click once on a script, or one of its assets, to change the focus to that script. VuGen applies any actions, such as clicking replay, to the script in focus.</p> <p>When any part of a script is selected, the menu options, toolbar and window panes display functionality relevant to the script's protocol. For example, if the script in focus is recorded in Web HTTP/HTML, the Recording Options button is displayed on the toolbar. However, if the script in focus is recorded in TruClient, the Develop Script button is displayed on the toolbar.</p> <p>Double-click the script's action to open it in the editor.</p> <p>You can drag and drop scripts (<scriptName>.usr) from the file directory to the Solution Explorer.</p>

Entity	Used for	Comments
Extra Files	Storing extra files that are used by the script.	<p>The data contained in extra files can include:</p> <ul style="list-style-type: none"> Common utility functions used by the script (for example, code)  You can add some types of extra utility function files to the parsing list to extend their functionality. For example, adding files enables "Go to Definition" functionality. See the "Options Dialog Box" on page 86. Definition of constants and variables used by the script (for example, code) Special assets used during script execution (such as .jpeg files) Data files manipulated by script code during script execution Additional files to be parsed. For details, see "Create and Open Vuser Scripts" on page 114. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> Example: The following are examples of valid file types that can be added as extra files:</p> <pre>.ws,.h,.c,.dat,.ini,.vbs,.java,.js,.txt,.tux,.rec,.msc,.vdf,.xml,.xsl,.dtd,.html,.htm</pre> </div> <p>You can drag and drop header files (<headerFileName.h> from your file directory. When you include files in the Extra Files node, these files are automatically included in a LoadRunner scenario.</p> <p>You can edit extra files in the editor if the file type is included in Tools > Options > Scripting Tab > Script Management. Double click the extra file to open it in the Editor. For details on how to modify the list, see "Scripting Options Tab" on page 96.</p>
Runtime Settings	Defining the way a Vuser script runs.	<p>You can access runtime settings for a specific script from the Runtime Settings node in the Solution Explorer > <Script> > Runtime Settings. For details, see "Runtime Settings Overview" on page 283.</p>
Parameters	Creating and managing parameters.	<p>You can access parameters for a specific script from the Parameters node in the Solution Explorer > <Script> > Parameters. For details, see "Parameterization Overview" on page 342.</p>

Entity	Used for	Comments
Replay Runs	Enables you to access the Replay Summary Reports for each iteration in the replay.	

Solution Explorer Structure and Context Menu Options

The following table lists the context menu options when working in the Solution Explorer.

UI Element and Description	Context Menu Options	Description
<Solution> Container for scripts.	Add New Script	Creates a new script and adds it to the solution. For details, see "Create a New Script Dialog Box" on page 128 .
	Add Existing Script	Adds an existing script to the solution.
	Save All Scripts	Saves the changes of all open scripts.
	Close Solution	Closes the current solution, along with its scripts, without saving the changes.
	Save Solution As...	Saves the solution with a new name or to a new location.

UI Element and Description	Context Menu Options	Description
<Script> Container for script assets including scripts actions, extra files, runtime settings and parameters.	Save Script	Saves the current script.
	Save Script As...	Saves the script with a new name or new location, such as ALM.
	Export to Template...	Saves the script as a template. For details, see "Create and Open Vuser Script Templates" on page 130 .
	Remove Script	Deletes a script from the solution or from the file directory, if you enable the check box.
	Select file to compare	Compares two files using the highlighted asset as the primary file.
	Compare to <filename>	Compares the primary file with the specified file.
	Compare to external file...	Compares a file to a file outside of the solution. This option sets the highlighted file as the primary file and opens Windows Explorer to enable you to select the secondary file.
	Select Folder to Compare	Compares one folder to another. The highlighted folder is the primary folder.
	Compare to External Folder...	Compares the selected folder and the primary folder.
	Open Script Folder	<p>When you record, VuGen creates a series of configuration, data, and source code files. These files contain Vuser runtime and setup information. VuGen saves these files together with the script.</p> <p>You can open the folder on the local disk where the script is saved. For scripts that are saved on a different storage location (such as ALM), this option opens the temporary folder on the local disk.</p>
























UI Element and Description	Context Menu Options	Description
<Actions> Container for individual actions including the default actions: <ul style="list-style-type: none"> • vuser_init • action • vuser_end 	Create New Action	Adds a new script action block to your script.
	Import Action	Imports an action from an existing script.
Extra Files Container for extra files associated with your script. You can access these extra files directly from VuGen.	Create New File	Creates a new file, and adds it to the Extra Files node of your script.
	Add Files to Script	Adds files to the Extra Files node of your script. The files that you add must already exist.
	Add Files from Folders and Sub-folders	Adds the contents of folders and sub-folders to the Extra Files node of your script.
Runtime Settings		Opens the runtime settings dialog box. For details, see "Runtime Settings Overview" on page 283 .
Parameters Enables you to create, edit, and list parameters associated with your script.	Parameters list	Opens the Parameter List dialog box. For details, see "Parameterization" on page 342 .
	Create new parameter	Opens the Select or Create Parameter dialog box.
	Edit parameter	Opens the Parameter Properties dialog box.
	Configure parameter delimiter	Opens the Parameter Delimiters Configuration dialog box.
Replay Runs Enables you to display the Replay Summary report.	Open Replay Summary	Opens the Replay Summary in the Editor for selected iteration.

See also:

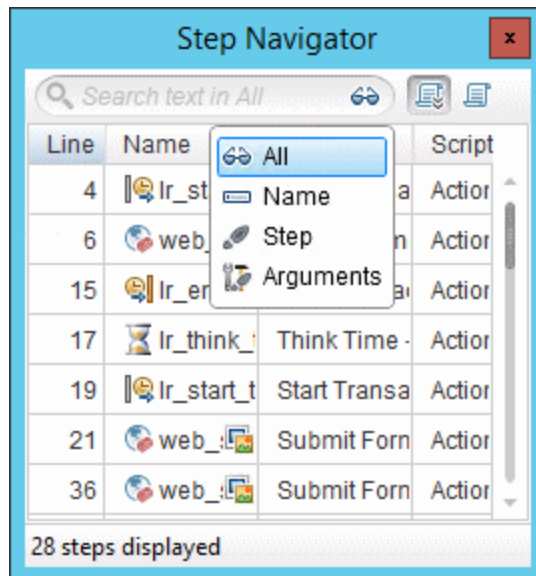
- ["VuGen Workflow" on page 113](#)

Step Navigator Pane

The Step Navigator is a table view of the API function calls in the script. In addition, the Step Navigator pane enables you to navigate to a selected step in your script.

Step Navigator ▼ 🔍 ✕		
🔍 Search text in All ↺ ↻ 📄		
Line	Name	Step
4	 web_convert_from_formatted	Convert Formatted Da
9	 web_add_auto_header	Service: Add Auto Hea
12	 web_url 	Url: cookies.html
23	 web_convert_from_formatted	Convert Formatted Da
29	 web_add_cookie	Service: Add Cookie
31	 web_convert_from_formatted	Convert Formatted Da
36	 web_add_cookie	Service: Add Cookie
38	 lr_think_time	Think Time - 30 (sec)
40	 web_url 	Url: cookies.html_2
49	 lr_think_time	Think Time - 6 (sec)
51	 web_url 	Url: CustomBodyThro
60	 web_convert_from_formatted	Convert Formatted Da
67	 lr_think_time	Think Time - 8 (sec)
69	 web_custom_request 	Custom Request: ech
81	 web_convert_from_formatted	Convert Formatted Da
86	 lr_think_time	Think Time - 13 (sec)
88	 web_custom_request 	Custom Request: ech
100	 web_convert_from_formatted	Convert Formatted Da



If your script contains many steps, you can use the search box to search for matching text in the different parts of the steps.



To access	View > Steps
Important information	<ul style="list-style-type: none"> You can search within an action or script but not across multiple scripts in a solution. You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40. You can view the script in either action or script scope. Every step that has a snapshot is marked with an icon. When hovering over a step that has an associated thumbnail it is presented as a tooltip. Double clicking a step, takes you to the corresponding location in the script and synchronizes all other panes. Step Navigation is synchronized based on the validity of the script. You can check the status of the pane during script editing.

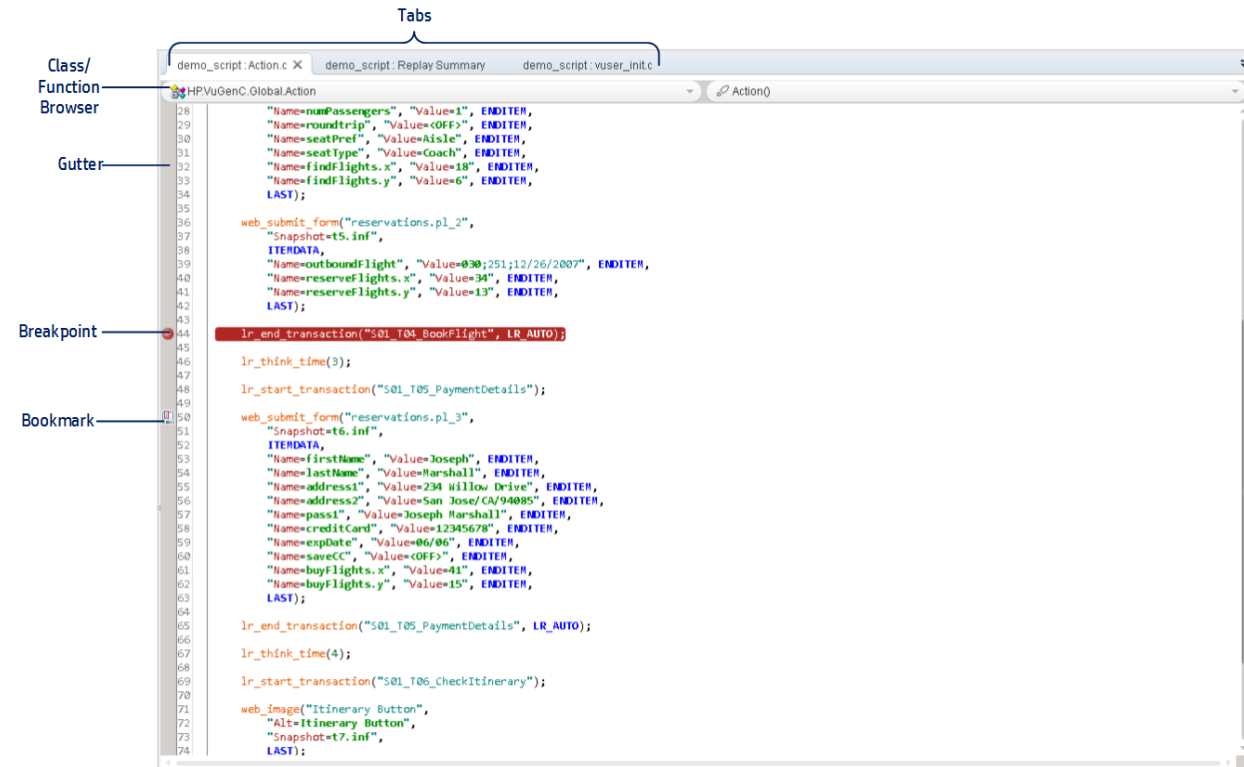
User interface elements are described below:

UI Element	Description
------------	-------------

Search box	<p>You can search different parts of the steps for matching text. The following parts of the steps can be searched:</p> <ul style="list-style-type: none"> • All. (Default) All parts of the step (name, step, arguments). • Name. The name of the step. • Step. The description of the step (for example, web_url, web_custom_request) • Arguments. The arguments for the step. <p>Enter the text you want to search for in your steps, in the search edit box and select the part of the steps you want to search. The Steps pane displays only those steps that match your search criteria.</p>
Line	The number of the step in the script.
Name	A step name.
Step	The step type.
 / 	File parser indicator. A green symbol indicates that parsing succeeded and a red symbol indicates that parsing failed.
Action	The action into which the step was created.
# steps displayed	Displays the total number of steps in the script or in the action.

Editor Pane

The Editor enables you to edit scripts and other related script files. In addition, you can open a browser session to search sites, such as the LoadRunner Forum.



To access	The editor is opened when VuGen is loaded.
Important information	<ul style="list-style-type: none"> • The editor is automatically displayed as part of the default layout. • Other interface panes, such as Output, Error, and Snapshot views, synchronize their displays based on your location in the editor. • Script modifications are displayed as highlighted text in the editor. For example: <ul style="list-style-type: none"> • Script has been modified but not saved. • Script has been modified and saved. • Breakpoint has been inserted.

Supported Programming Languages

Recorded scripts are generated in C, which has full language and parsing support in VuGen. You can enhance your Vuser scripts by adding standard ANSI C functions. ANSI C functions allow you to add comments, control flow statements, conditional statements to your Vuser scripts. You can add standard ANSI C functions to any type of Vuser script. For details, see ["C Vuser Scripts" on page 836](#).

In addition, the VuGen editor enables you to write manual scripts with the following programming languages:

- Java: For details, see ["Java Vusers" on page 837](#).
- C#: For details, see [".NET Vusers" on page 837](#).
- VB.NET: For details, see ["Create a Vuser Script in Visual Studio" on page 840](#).





Tips for Working with the Editor



- Press **Ctrl + Tab** to display a list of tabs and panes. Highlight and click to switch tabs in the editor.
- Press **Ctrl + Tab** to display a list of tabs and panes. Highlight and click to switch tabs in the editor.
- Press **Ctrl + G** or select **Search > Go to...** to go to a specific line in the script.
- Double-click an asset in the Solution Explorer to open it in the editor.
- Drag any type of file into the editor and modify it. To save your changes, select **File > Save File As**.

Code Completion and Tooltips

Code completion (autocomplete) enables you to quickly and accurately write code by providing a list of code items from which you can select options. Press **CTRL + SPACE** to activate statement completion when your cursor is in the editor. Tooltips appear when you hover over a code element with your mouse.

The following table describes available code completion items, scope, identifying icon, and tooltip context:

Code Completion Item	Scope	Icon	The Tooltip Displays
ANSI C Keywords and Types	All possible standard C keywords.		Function type, name, and parameters
LR API Functions	All steps in the script.		LR API Step
LR API Constants	Used to delimit groups of parameters in steps. For example, ENDITEM		LR API Constant
User Functions	All the functions that you have defined in action files.		<ul style="list-style-type: none"> • Function type, name and parameters. • When Using Functions (Method Insight) The required arguments, highlighting each argument as you define it moving to the next argument when you enter the delimiter.

Code Completion Item	Scope	Icon	The Tooltip Displays
Variables	<p>Local variables – visible only in the function where they have been defined.</p> <p>Global variables – defined outside of any function body. Available everywhere in the script.</p>		Type and name.
Parameters	Available only in the function body where they have been defined.		<ul style="list-style-type: none"> • Parameter type and name • When Using parameters (Method Insight) <p>The required arguments, highlighting each argument as you define it moving to the next argument when you enter the delimiter.</p>

By default, VuGen uses code completion globally. To disable code completion, select **Tools > Options > Editor Tab > Code Completion**. Clear the **Enable code completion features** check box.

Code-Coloring

To facilitate script writing and debugging, code item types are colored by an identifying background and foreground. The colored text enables you to easily read scripts and scan for syntax errors.

The following table provides examples of code item types and their assigned colors.

Code Type	Color Example
Comments	<code>/* comment */ // comment</code>
Keywords	<code>if (a) { } else { }</code>
Method Parameter Name	<code>foo("parameter=value")</code>
API function	<code>web_url</code>
Method Call	<code>foo()</code>
String	<code>char * text = "Hello, World!"</code>

In addition, you can customize code item types to suit your needs by selecting **Tools > Options > Editor Tab > Code Color**.

Script Folding

Script folding enables you to selectively hide and display sections of a script, making it easier to manage large scripts by viewing only those sections that you are currently editing. For details, see ["Editor Options Tab" on page 90](#).





Community Search

You can perform Web searches from the VuGen toolbar, which opens a browser tab in the editor. The default Web site is the LoadRunner Forum which enables you to search topics, post questions, or blog about your expertise. You can add additional search sites by selecting **Tools > Options > General > Community**. For details on adding additional sites, see ["General Options Tab" on page 86](#).

User Interface Elements

The following table describes the editor user interface and the right-click (context) menu options (unlabeled elements are shown in angle brackets).

UI Element	Description
<Gutter>	The editor gutter enables you to add toggle functionality including: <ul style="list-style-type: none">• Breakpoints For details, see "Working with Breakpoints" on page 317 or "Breakpoints Pane" on page 78.• Bookmarks For details, see "Use Bookmarks" on page 309.
Class/Function Browser	When enabled, you can navigate quickly to a specific class or function in your script by selecting it from the drop-down list in the browser.
Line numbers	Display of line numbers in the script.
Context Menu	
Comment or Uncomment	Enables you to comment or uncomment highlighted script lines.
Decode with DFE	Applies to Web - HTTP/HTML including Web - HTTP/HTML steps in Flex Vuser scripts. For details see, "Data Format Extensions (DFEs) - Overview" on page 704 .
Show Snapshot	Shows the snapshot associated with the highlighted script step.
Show Arguments	When the cursor is placed on a function, the Step Properties dialog box appears with the details of the function's arguments.

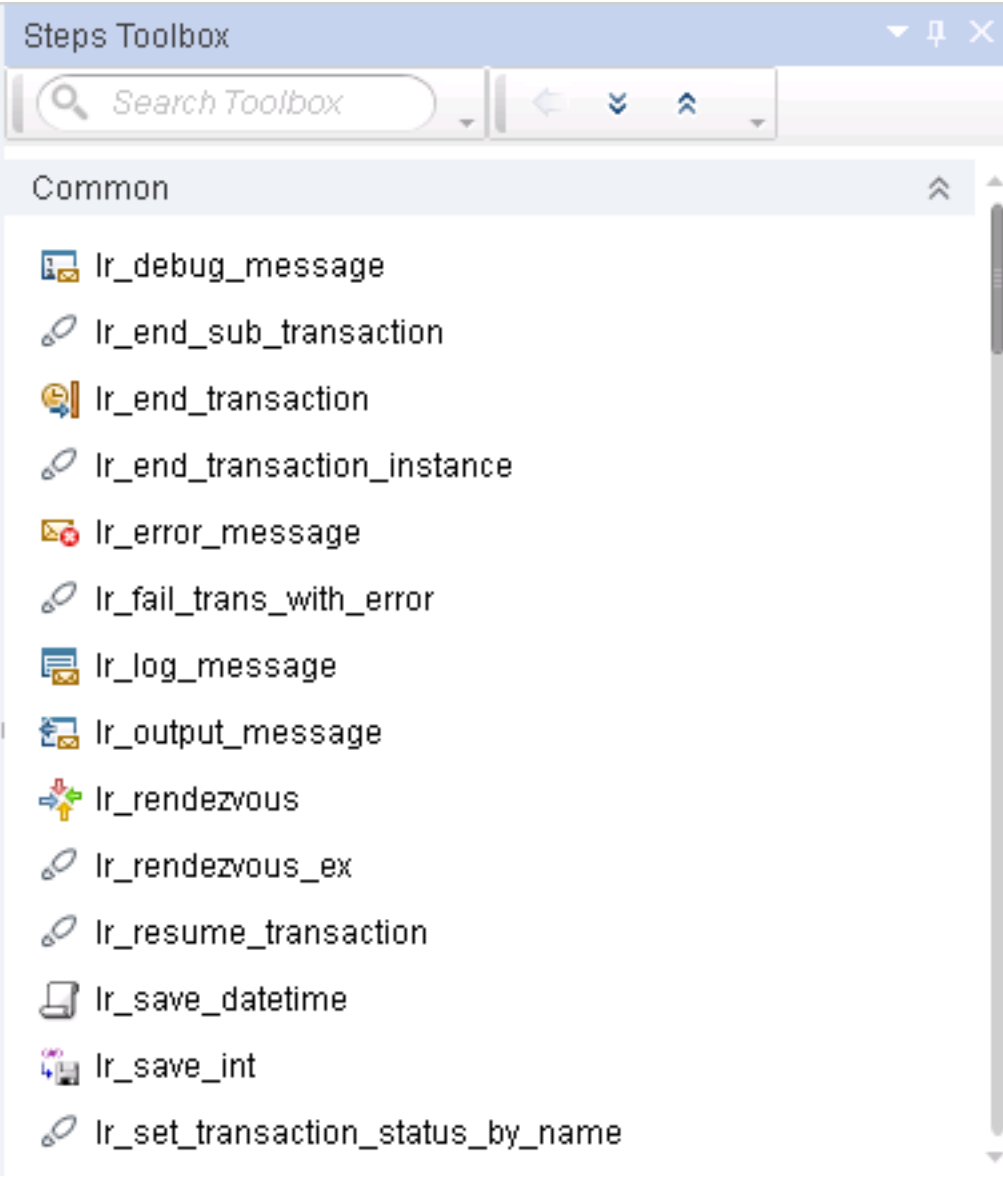
UI Element	Description
Correlate Selection	Opens Design Studio and scans for dynamic values to be correlated in the selected script section.
Restore Masked String (value)	Restores the original value for a masked string.
Mask String	When the cursor is placed in a password, for example, inserts an <code>lr_unmask</code> function to encode the string.
Surround with Transaction	Inserts an <code>lr_start_transaction</code> function immediately above the highlighted script and a <code>lr_end_transaction</code> function immediately below the highlighted script.
Go to Step in Replay Log	Navigates to the location in the Output pane that correlates to the function in the editor.
Insert	 Inserts a New Step into your script.  Inserts a Start Transaction step into your script. For details, see " Transaction Overview " on page 323.  Inserts an End Transaction step into your script. For details, see " Transaction Overview " on page 323.  Inserts a Rendezvous point step into your script. For details, see " Rendezvous Points " on page 328. Comment. Inserts comments into your script. Log Message. Inserts an <code>lr_log_message</code> step into your script.
Toggle Breakpoints	Adds or removes a breakpoint.
Search Community	Opens the default Community Search browser.

 See also:

- "[Editor Options Tab](#)" on page 90

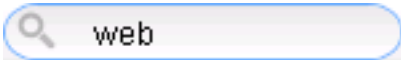


Steps Toolbox Pane

This pane enables you to drag and drop API functions into your script.

UI example	
To access	<p>Use one of the following:</p> <ul style="list-style-type: none">• Design > Insert in Script > New Step• Click within a script and select Insert > New Step from the right-click menu• Press Ctrl+Alt+B

Important information	<ul style="list-style-type: none"> • A step's associated parameter dialog box opens when you add the step to the script. • You can drag and drop steps into your script. • You cannot drag and drop a step into a step from the Steps Toolbox but you can manually add a step parameter within a step. • If you insert a step into the incorrect location in your script, the script may fail.
Relevant tasks	"Insert Steps into a Script" on page 340.

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
<Function List>	<p>Displays a list of available functions divided into the following categories:</p> <ul style="list-style-type: none"> • Common • Web checks • Services • XML • Web JS • Async
	Search. Enables you to perform an incremental search in the function list by entering text. For example, if you type "web" into the search box, the function list will display only those function that include the letters "web".
	Add. Add the highlighted step to the current location in your script.
	Expand/Collapse. Expand or collapse the step categories.






Bookmarks Pane

The Bookmarks pane displays a list of the bookmarks in your Vuser script. You can navigate between the bookmarks to help analyze and debug your code.

To access	View > Bookmarks
------------------	----------------------------

Important information	<ul style="list-style-type: none"> • All bookmarks added to a Vuser script are maintained after you close and reopen the Bookmarks pane. • You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.
Relevant tasks	"Use Bookmarks" on page 309

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Bookmarks list>	Displays a list of all bookmarks that are defined in the Vuser script. You can double-click any bookmark line to navigate directly to the relevant line in the Vuser script.
	Toggles the status of the selected bookmark.
	Navigates to previous bookmark in the pane.
	Navigates to next bookmark in the pane.
	Deletes the selected bookmark.
	Deletes all bookmarks.

See also:


- ["VuGen User Interface" on page 37](#)

Snapshot Pane

Snapshots contain the data generated by the traffic between the client and the server and are captured when a script is recorded and when the script is replayed.






Snapshots are displayed in various formats and provide different functionality depending on the Vuser protocol.




Note: The Snapshot pane is not available for all Vuser protocols.

To access	Select View > Snapshot , or click the Show Snapshot Pane button  on the VuGen toolbar.
Important information	<ul style="list-style-type: none"> • A snapshot displays all data associated with a specific step in a script. • Snapshots enable you to search for correlations, compare record versus replay snapshots and search for the specific values using the standard search operation. • Press Ctrl+F to search for text within a snapshot view. (Search is not supported for the JSON view.) • The appearance and functionality of the Snapshot pane varies depending on the protocol of the current Vuser script. In addition to the standard controls, the Snapshot pane may display controls that are specific to the current Vuser protocol. • The new Web snapshot model is backward compatible with previous versions of LoadRunner. However some snapshot data may be missing. If this occurs, regenerate the script. • When using Windows 2008 R2 and opening a snapshot from the step navigator in SAP GUI and Web protocols, the snapshots might not open automatically. Workaround: Internet Explorer Enhanced Security Configuration must be disabled to view help content. It is enabled by default. (Control Panel > Administrative tools > Server manager > Configure IE ESC). • You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.

Standard Snapshot pane controls

The Snapshot pane displays the following standard controls:

UI Element	Description
 Single	Shows a single snapshot in the Snapshot pane.
 Split	Splits the Snapshot pane to show two snapshots.
 Horizontal	Displays two snapshots in the Snapshot pane - one to the side of the other.
 Vertical	Displays two snapshots in the Snapshot pane - one above the other.
 Recording	Shows the record snapshot in the Snapshot pane.

	Shows the replay snapshot in the Snapshot pane.
Iteration: 1 ▾	Select the iteration number of the replay snapshot to display.
	<p>Synchronizes the display of the two snapshots if the Snapshot pane is split.</p> <p>Note: Snapshot synchronization is available for only specific Vuser protocols, and for only specific views within the protocols.</p>
	<p>Compares the two snapshots that are currently displayed in the Snapshot pane. To enable the Compare functionality, you must first split the Snapshot pane to show two snapshots. By default, VuGen uses the <i>WDiff</i> utility to compare snapshots. You can specify an alternative comparison tool as described in "Scripting Options Tab" on page 96.</p> <p>Note: The snapshot comparison functionality is available for only the Web HTTP/HTML and Web Services protocols.</p>

Snapshot pane controls for Citrix, RDP, and SAP protocols

User interface elements are described below:

UI Element	Description
Image	Displays a graphical representation of the snapshot. You can synchronize the display of two snapshots in the Snapshot pane. Snapshots display faster when the Image view is used than when the Full view is used.
Full	Displays a graphical representation of the snapshot. You cannot synchronize the display of two snapshots in the Snapshot pane. Snapshots display slower when the Full view is used than when the Image view is used.
Context Menu	
Copy Image to the Clipboard	Copies the image to the clipboard.
Insert Mouse Click	Inserts a mouse click function, for example, <code>ctx_mouse_click</code> , at the point of the cursor in the script.

Insert Mouse Double Click	Inserts a double mouse click function, for example, <code>ctrx_mouse_double_click</code> , at the point of the cursor in the script.
Insert Sync on Bitmap	Inserts a sync on image function, for example, <code>ctrx_sync_on_bitmap</code> , at the point of the cursor in the script. Select the bitmap area with the cursor to get the bitmap values (x,y coordinates, width, height) for the function.
Insert Sync on Bitmap (by coordinates)	Inserts a sync on image function with synchronization area coordinates as function parameters, for example, <code>ctrx_sync_on_bitmap</code> . The parameters are entered in a dialog box displayed after selecting this option. The function is inserted at the point of the cursor in the script.
Insert Get Bitmap Value	Inserts a get bitmap value function, for example, <code>ctrx_get_bitmap_value</code> , at the point of the cursor in the script. This function retrieves the hashed string value of a bitmap for use in custom synchronization functions. Select the bitmap area with the cursor to get the bitmap values (x,y coordinates, width, height) for the function.
Insert Get Text	<p>Inserts a get text function, for example, <code>ctrx_get_text</code> function, at the point of the cursor in the script. Select the text area with the cursor to get the values (x,y coordinates, width, height) for the function.</p> <div> <p>Note:</p> <ul style="list-style-type: none"> Available only when the Vuser script is recorded on a Citrix server, with the LoadRunner Citrix agent installed on it. The Citrix agent does not support this function on Windows 10, Windows Server 2016, or later versions of Windows. </div>
Insert Sync on Text	<p>Inserts a sync on text function, for example, <code>ctrx_sync_on_text_ex</code> function, at the point of the cursor in the script. Select the text area with the cursor to get the values (x,y coordinates, width, height) for the function. This function waits until the specified text is displayed (agent installations only).</p> <div> <p>Note:</p> <ul style="list-style-type: none"> Available only when the Vuser script is recorded on a Citrix server, with the LoadRunner Citrix agent installed on it. The Citrix agent does not support this function on Windows 10, Windows Server 2016, or later versions of Windows. </div>

Insert Obj Get Info	<p>Inserts an object get info function, for example, <code>ctrx_get_obj_info</code>, at the point of the cursor in the script. This function retrieves the current state of the requested object property.</p> <p>Note: This menu item only appears when the Vuser script is recorded on a Citrix server, with the LoadRunner Citrix agent installed on it.</p>
Insert Sync on Obj Info	<p>Inserts a sync on obj info function, for example, <code>ctrx_sync_on_obj_info</code>, at the point of the cursor in the script. This function causes VuGen to wait for a certain state before continuing.</p> <p>Note: This menu item only appears when the Vuser script is recorded on a Citrix server, with the LoadRunner Citrix agent installed on it.</p>

Snapshot pane controls for Windows Sockets protocol

The snapshot for Windows Sockets protocol are displayed as textual and hexadecimal representations of data buffers sent and received.

User interface elements are described below:

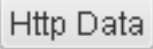
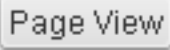


UI Element	Description
Hex	Displays the buffer data in hexadecimal.
Text	Displays the buffer data as text.
Go To	Opens the Go To Offset dialog box that enables you to navigate within the data buffer.
Context Menu	
Create Correlation (Select the text for correlation, from the Response Body tab)	Opens the Correlation dialog box. For details see "Correlating" on page 232 .

Snapshot pane controls for Web - HTTP/HTML protocol

The snapshot for Web - HTTP/HTML protocol are displayed as textual and hexadecimal representations of the request and response.

User interface elements are described below. These are available for both the Recording and Replay snapshots.

UI Element	Description
------------	-------------




	Displays step data in HTTP format. This enables you to view in-depth information about the step, including request data, response data, cookies, and headers. For more information, see "Web Snapshots - Overview" on page 697 .
	Displays step data in HTML format.
	<p>(Http Data view only) Displays the HTTP flow data in a tree structure.</p> <p>The data is separated into separate tabs: Raw Data, Request Body, Response Body, Headers, Cookies, and Query String.</p> <p>For some of the above tabs, you can display the data in text, hexadecimal, or json views, using the buttons on the right: Text View, Hex View, and JSON View.</p> <p>Note: Depending on your application, some tabs may not have any data.</p>
	(Http Data view only) Displays the HTTP flow data in a list format.
Iteration	The iteration number to display in the pane (only for Replay snapshots)
Context Menu	
Create Correlation	<p>Opens the Correlation dialog box. For details see "Correlating" on page 232.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p>
Create Correlation Rule	<p>Opens the Add as Rule dialog box. For details see "Manually Correlate Scripts" on page 242.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p>
Create Parameter	<p>Opens the Create Parameter dialog box. For details see "Create Parameters" on page 347.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p>
Search Community	Opens the LoadRunner Support Forum.

Add Text Check Step	<p>Opens the Find Text dialog box. For details see "Add a Text Check From the XML View in the Snapshot Pane" on page 282.</p> <div> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p> </div>
---------------------	---

Snapshot pane controls for protocols with an XML request/response (such as Web Services)

The snapshots for XML protocols are displayed as XML and textual and hexadecimal representation of the request and response bodies.

User interface elements are described below:


UI Element	Description
 Response	Displays an XML view of the server response.
 Request	Displays an XML view of the request.
	Opens the XPath search dialog box, allowing you to search the XML code using a standard XPath expression.
Context Menu	
Copy Selection	Copies the text that is selected in the text view to the clipboard
Search Community	Performs a community search using the text that is selected in the text view as the search string. For details about performing a community search, see "Editor Pane" on page 54.
Copy XPath	In the tree view, copies the XPath of the selected node to the clipboard. In the text view, copies the XPath of the XML element in which the cursor is located to the clipboard.
Copy full value	In the tree view, copies the full XML code of the selected node to the clipboard. In the text view, copies the full XML code of the XML element in which the cursor is located.
Node Properties	Opens the XML Node Properties dialog box.

Insert XML Check	<p>Opens the Insert XML Check dialog box that enables you to insert an XML Find step into the Vuser script.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p>
Save value in parameter	<p>Opens the Save Value as Parameter dialog box that enables you to save the selected value to a simple parameter.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p>
Save XML in parameter	<p>Opens the Save Value as Parameter dialog box that enables you to save the selected value to an XML parameter.</p> <p>This option is available in the Response view only.</p>
Create Correlation	<p>Opens the Correlation tab in the Design Studio. The text selected in the Snapshot pane appears as a manual correlation entry in the Design Studio. For details, see "Manually Correlate Scripts" on page 242.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes in the tree view, and when text is selected in the text view.</p>
Create Correlation Rule	<p>Opens the Add as Rule dialog box that enables you to add the selected text as part of a correlation rule. For details, see "Correlation Tab [Design Studio] Overview" on page 234.</p> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes in the tree view, and when text is selected in the text view.</p>
Copy XML	

Snapshot pane controls for Database Protocols

User interface elements are described below:

UI Element	Description
------------	-------------

 Recording	Shows the record snapshot in the Snapshot pane.
Data Grid	Displays data in a data grid.
Context Menu	
Create Correlation	<p>Opens the Correlation dialog box. For details see "Correlating" on page 232.</p> <div> <p>Note: This option is available in the Response view only. This option is available for attribute and value nodes only.</p> </div>
Save Grid to File	Saves the grid's contents to a CSV file.

Copying images to the clipboard

You can copy an image-based snapshot to the clipboard. This enables you to import the image into a graphics application, where you can analyze and modify the graphic.

For details on how to copy a snapshot to the clipboard, see ["Work with Snapshots" on page 279](#).

Note: The "copy snapshot to the clipboard" functionality is available for only RDP, Citrix, and SAP GUI protocols.

Copying snapshot text to the clipboard

You can copy text from a snapshot to the clipboard. You can then paste the text from the clipboard into another application.

For details on how to copy snapshot text to the clipboard, see ["Work with Snapshots" on page 279](#).

Note: The "copy snapshot text to the clipboard" functionality is available only for Ajax - Click & Script protocol.

Customized Snapshot pane functionality

In addition to the basic Snapshot pane functionality, the Snapshot panes for some Vuser protocols include customized functionality. For example, the Snapshot pane for RDP Vuser scripts lets you display snapshots in either **Full** or **Image** modes; the Snapshot pane for Winsock Vuser scripts lets you display snapshots in either **Text** or **Hex** modes. The controls for the customized functionality can be found in the Snapshot pane toolbars.

Activating the Snapshot on error functionality

In addition to showing record and replay snapshots, the Snapshot pane can display snapshots of errors that occurred during the replay of a script. The "snapshot on error" functionality is available for only specific Vuser protocols.

You can generate and display snapshots of errors only if the "snapshot on error" functionality is activated.

To activate the snapshot-on-error functionality:


1. Click **Replay > Runtime Settings**. The runtime settings dialog box opens.
2. Under **General**, click **Miscellaneous**.
3. Under **Error Handling**, select the **Generate snapshot on error** check box.

See also:



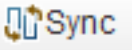






- ["Work with Snapshots" on page 279](#)

Thumbnail Explorer

This pane enables you to flip through thumbnail images of your business process, enhancing your ability to navigate to specific locations in the **Editor** based on a visual representation of a step. Conversely, you can scroll through the **Editor** and see the visual context of your script in the **Thumbnail Explorer**.

To access	Use one of the following: <ul style="list-style-type: none">• View >Thumbnail Explorer• Click the  button on the VuGen toolbar.
Important information	<ul style="list-style-type: none">• You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.• You can configure the Thumbnail Explorer in Options > Scripting Options. For details, see "Scripting Options Tab" on page 96.• Thumbnails are created in the same order as the actions in the Solution Explorer and not controlled by the settings in runtime settings > General > Run Logic.• Enabling automatic creation of thumbnails in Options > Scripting > Thumbnails enables VuGen to create thumbnails during the application's idle time.• If the Windows Aero theme is enabled, thumbnails capture more realistic images of the application.
Relevant tasks	"How to Modify the VuGen Layout" on page 40

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
 Goto Step	Moves the cursor to the step in the Editor associated with the highlighted thumbnail in the Thumbnail Explorer .
 Full Screen	Enables a full screen view of the thumbnail.
 Sync	Synchronizes the scrolling in the Editor with the associated thumbnail in the Thumbnail Explorer and step in the Step Navigator .
	Filters out minor thumbnails that are not directly related to the recorded business process.
	Refreshes the generated thumbnails.
	Scroll a page left in the Thumbnail Explorer .
	Move to the previous thumbnail in the Thumbnail Explorer .
	Move to the next thumbnail in the Thumbnail Explorer .
	Scroll a page right in the Thumbnail Explorer .




Errors Pane

The Errors pane lists the replay and syntax errors found in your script, and enables you to locate each error so that you can resolve it.

To access	View > Errors
------------------	-------------------------

Important information	<ul style="list-style-type: none"> • After every test process, such as code generation and replay, you can check the error pane for the error log. • The error log includes errors, warnings and messages. • Community search is available with context menu on highlighted error. • Double-click message to jump to the location in the script. • You can move this pane to different areas of the main user interface. For details, see "How to Modify the VuGen Layout" on page 40. • When you open an existing script, the items displayed in the Error Pane are from the latest replay or compilation.
------------------------------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Define Available Categories> dropdown	Filters the error list by the source of the error.
 Errors: 0	Shows or hides syntax errors.
 Warnings: 0	Shows or hides warnings detected during the run.
 Messages: 0	Shows or hides informational messages detected during the run.
! <Exclamation Point>	Task type: <ul style="list-style-type: none"> • Error • Warning • Informational message
Line	The line containing the error.

Description	<p>Description of the error, warning or message and advice on how to fix the problem. For example, a syntax error is displayed if you opened a conditional block with an If statement but did not close it with an End If statement, the description is Expected Expression.</p> <div> <p>Note: If the description does not fit within the Description column, a tooltip displays the full description when you hover the cursor over the column. In certain cases, VuGen is unable to identify the exact error and displays a number of possible error conditions, for example: Expected 'End Sub', or 'End Function', or 'End Property'. Check the statement at the specified line to clarify which error is relevant in your case.</p> </div>
File	The name of the file that contains the problematic statement.
Path	The full path of the file that generated the error.
Test	The name of the script in which the error was detected.

See also:

- ["VuGen User Interface" on page 37](#)
- ["Debugging" on page 313](#)

Tasks Pane




This pane enables you to add, edit and track tasks associated with an individual script or the overall goals of the project. Tasks are divided between user defined tasks and tasks that are inserted into the Vuser script as action items using keywords such as ToDo, Undone and FixMe.

User defined tasks are displayed when you select the User Tasks option in the Task pane. Action items are displayed when you select the Comments option in the Task pane.

To access	View > Tasks
See also	"VuGen User Interface" on page 37

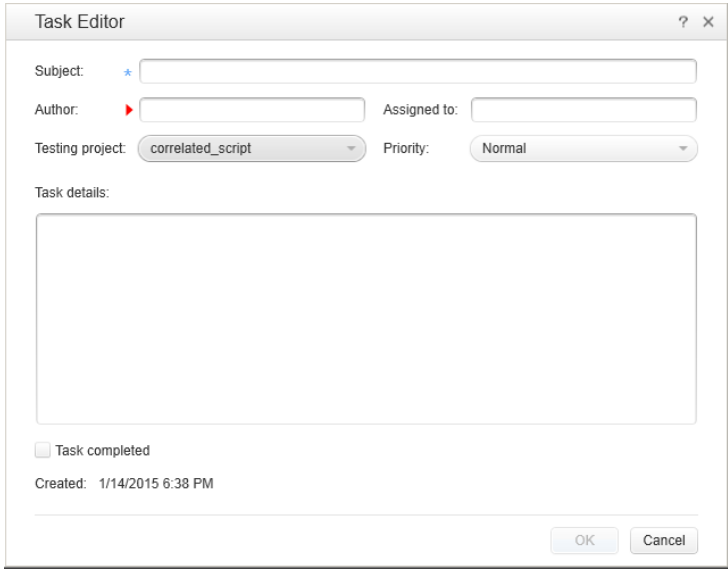


User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
------------	-------------

Comments view	<p>Comment tasks are added directly into your script using the comment syntax of your scripting language and include a keyword such as TODO or FIXME. For example, in C a comment script would look like this:</p> <pre>//TODO Add Parameter</pre> <p>The Task Pane displays the following information about each task:</p> <ul style="list-style-type: none">• !: Task Type• Line: What line the task is located. Double-clicking the task jumps to that location in the script.• Description: The Keyword and the task contents.• File: Action• Path: File location of the action.
User Tasks view	<p>You can add, edit, delete user tasks:</p> <p>Click  to add a task with the "Task Editor" below.</p> <p>Click  to edit a task with the "Task Editor" below.</p> <p>Click  to delete a task.</p>
<Task Filter>	<p>You can filter tasks associated with a particular script or see all the tasks associated with the solution.</p>

Task Editor

This dialog box enables you to add or edit user tasks.

UI example	
To access	<p>View > Tasks > User Tasks tab</p> <p> Adds a new task</p> <p> Opens the highlighted task for editing</p>
Important information	<ul style="list-style-type: none"> • A blue asterisk indicates a required field. • You can copy and duplicate tasks using the right-click menu.


User interface elements are described below:

UI Element	Description
Subject *	Brief description of task. Required field.
Author *	User who initiates the task
Assigned to	User who is assigned to complete the task
Testing Project	Script associated with the task.
Priority	<p>You can assign a priority to the task:</p> <ul style="list-style-type: none"> • High • Normal • Low
Task details	A textual description of the task

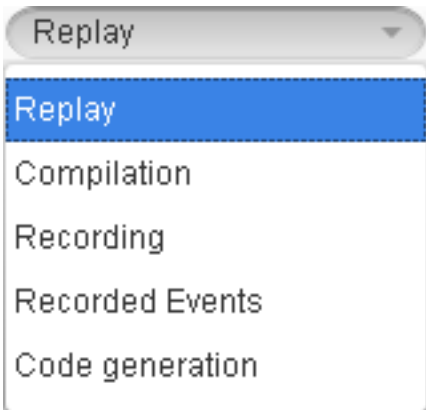
UI Element	Description
Task completed	A check box indicating that you have completed the task
Created	A non-editable field that displays the date and time the task was created








Output Pane

The Output pane displays messages that were generated during the recording, compilation, and replay of your script.

To access	Select View > Output or click the Show Output Pane button  on the VuGen toolbar.
Important information	<ul style="list-style-type: none"> You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40. When you open an existing script, the items displayed in the Output Pane are from the latest replay or compilation.
See also	"VuGen User Interface" on page 37

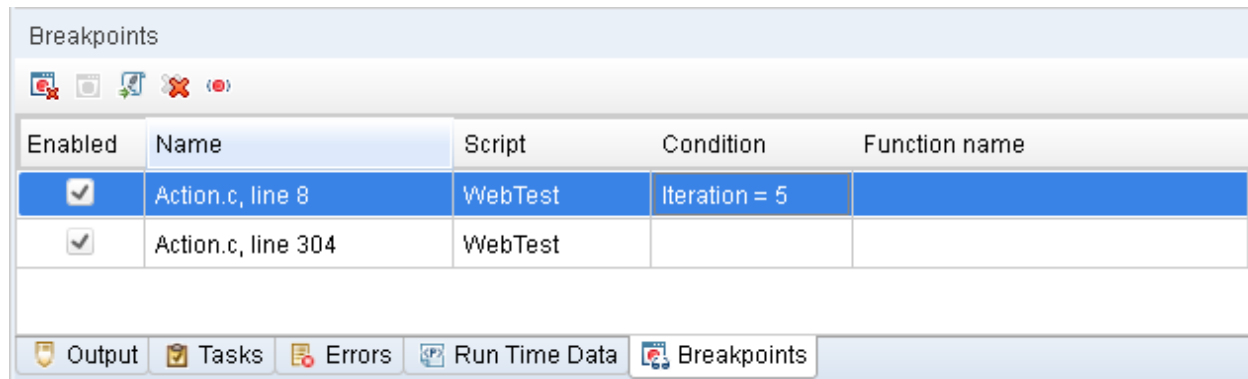
User interface elements are described below:

UI Element	Description
	<p>The type of output to display. The following types are available:</p> <ul style="list-style-type: none"> Replay. Displays the messages generated by the script replay. <ul style="list-style-type: none"> Note: If you double-click an entry in the Replay log, VuGen moves the cursor to the corresponding line in the Editor. Compilation. Displays the compilation messages. Code Generation. Displays the code generated during the recording. Recording. Displays the messages generated during the recording. Recorded Events. Displays events that occurred during recording.

UI Element	Description
	Clears all of the messages from the message list.
	Toggle Line Wrap. When selected, wraps the text of each message onto the next line - as required.
	Jumps to the location in the source document relevant to the selected output message.
	<p><Find box>. The text string that you want to find. You can refine your search by selecting one of the Options described below.</p> <p>Press ENTER to begin the search.</p>
	<p>Find Previous / Find Next. Highlights the next or previous string that matches the text you entered in the Find box.</p> <p>These buttons are available only after you enter text in the Find box.</p>
	<p>Enables you to refine your search with the following options:</p> <ul style="list-style-type: none"> • Match Case. Distinguishes between upper-case and lower-case characters in the search. • Match Whole Word. Searches for occurrences that are only whole words and not part of longer words. • Use Regular Expression. Treats the specified text string as a regular expression. <p>Note: Extended regular expressions and multi-line searches are not supported.</p>
	Opens the Save As dialog box, enabling you to save the contents of the message list as a text file.
View Summary [Available in the Replay log only]	Opens the Replay Summary tab. For details, see "Replay Summary Pane" on page 108 .

Breakpoints Pane

The Breakpoints pane enables you to set and manage breakpoints to help analyze the effects of the script on your application at pre-determined points during script execution.



To access	View > Debug > Breakpoints
Important information	You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40 .
Relevant tasks	"Debug Scripts with Breakpoints" on page 320

Breakpoints Pane

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Removes the selected breakpoint.
	Removes all breakpoints.
	Locates the cursor in the Vuser script at the line that contains the selected breakpoint.
	Disables the selected breakpoint if it is enabled, and enables the selected breakpoint if it is disabled.
	Allows you to enter conditions for the selected breakpoint. See "Breakpoint Condition Dialog Box" on the next page for more details.
<Breakpoints Grid>	A list of the breakpoints and their locations in the script. To enable a breakpoint, select the Enable check box next to that breakpoint. To disable a breakpoint, clear the Enable check box.
Enabled	A check box that specifies whether the breakpoint is enabled or disabled, and enables you to enable or disable the adjacent breakpoint.

Name	The name of the file that contains the breakpoint, and the line number within the file that contains the breakpoint.
Script	The name of the Vuser script that contains the breakpoint.
Condition	The condition that applies to this breakpoint. If there is no condition, the replay will always stop at the breakpoint.
Function Name	The name of the function within the Vuser script that contains the breakpoint.

Breakpoint Condition Dialog Box

The Breakpoint Condition dialog box enables you to condition breakpoints by the iteration number, the value of one or more parameters, or a combination of both parameter values and iteration number.

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
Break when iteration number is	A check box that enables you to specify a breakpoint dependant on the iteration number.
<Operand>	Choose one of the following operands: <ul style="list-style-type: none"> • = (equal) • <= (less than or equal to) • => (equal to or greater than) • < (less than) • > (greater than)
to	Enter an iteration number.

Break when Parameter	A check box that enables you to specify a breakpoint dependant on the value of a parameter.
<Parameter Name>	Choose a parameter from the drop-down list
Value is	Enter the parameter value for which you want a breakpoint.
OK	Apply the conditions to the selected breakpoint. All future replays will only stop at this breakpoint if these conditions are met.

Call Stack Pane

This debug pane enables you to view information about the functions that are currently on the call stack of your script.

To access	View > Debug > Call Stack
Important information	<ul style="list-style-type: none"> • This pane is relevant only when a run session is paused. • You can double-click any element in the Call Stack pane to navigate to the beginning of the relevant function/action. • This pane is read-only. • You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.
See also	<ul style="list-style-type: none"> • "VuGen User Interface" on page 37 • "Debugging" on page 313

User interface elements are described below:

UI Element	Description
Function name	The name of the function currently called.
File name	The name of the file containing the called function.
Line #	The line number on which the function definition begins.





Watch Pane

The Watch pane enables you to monitor variables while a script runs.

To access	View > Debug > Watch
------------------	-----------------------------------

Important information	<ul style="list-style-type: none"> This pane is relevant only when a run session is paused. You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.
------------------------------	---

User interface elements are described below:

UI Element	Description
	Enables you to add a variable to the watch list.
	Enables you to edit the selected variable in the watch list.
	Deletes the selected variable from the watch list.
	Deletes all the variables from the watch list.
Expression	The variable whose value you want to watch.
Value	The current value of the variable. The evaluated value is displayed only when a run session is paused.
Type name	The type of the variable's value after it is evaluated (for example, Integer or Char). If a variable cannot be evaluated in the current context, the type displayed is Incorrect expression .

Runtime Data Pane

The Runtime Data pane displays information about the current script execution.

To access	View > Debug > Runtime Data
Important information	<ul style="list-style-type: none"> The Runtime Data pane is accessible during script replay only. You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.
See also	"Replay a Vuser Script" on page 278


User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
------------	-------------

Iteration	Displays the current iteration number.
Action	Displays the Action name of the currently replayed step.
Line Number	Displays the line number of the currently replayed step.
Elapsed time	Displays the time that has elapsed since the start of the replay.
<Parameters>	Displays all parameters defined, together with the script and their substitution values based on the selected update method (sequential, unique, etc.). VuGen shows this information even if the parameter is not used in the script.



.NET Recording Filter Pane




This pane enables you to manage .NET filters.

To access	View > .NET Recording Filter or the Show .NET Recording Filter toolbar button  .
See also	<ul style="list-style-type: none"> • ".NET Filters Overview" on page 583 • ".NET Filters - Advanced" on page 587 • "Guidelines for Setting .NET Filters" on page 584
















User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Filter List>	<p>A list of the recording filters.</p> <ul style="list-style-type: none"> • Environments. Built-in system filters for .NET Remoting, ADO.NET, Enterprise Services, and WCF (Windows Communication Foundation). • Custom filters. Filters that you created manually for this script. TO create a custom filter, click New.
<Filter Tree>	<p>The Filter tree uses symbols to illustrate the elements and their status. For details about each of the icons, see the table below.</p> <ul style="list-style-type: none"> • Element icons represent the type of element—assembly, namespace, class, method, structure, property, events, or interfaces. • A check mark or X adjacent to the element icon, indicates whether or not the element is included or excluded. • A bold element indicates that it was explicitly included or excluded. This may be a result of being manually included or excluded by the user or by a pre-defined rule in the environment filter. If you reset a bold node, it returns to its original, non-bold state.

New	Opens the Create a New Filter dialog box, in which you create an empty filter or a new filter based on an existing one. For more information, see " Create a New Filter Dialog Box [.NET Protocol] " on page 589.
Save	Saves the changes you made to filter.
Delete	Deletes the selected custom filter. The filter pane prompts you for a confirmation.
Add Reference	<p>Opens the Add Reference dialog box with a list of .NET Framework components or assemblies in the Public Assemblies folder. For more information, see "Add Reference Dialog Box [.NET Protocol]" on page 589.</p> <div> <p>Note: If you add a reference to a DLL for a .NET filter, VuGen adds it to the script's reference list only if the script accesses the DLL's methods during recording.</p> </div>
Remove Reference	Removes the assembly that is selected in the Filter pane and disables all of the rules associated with it. All disabled rules are enabled after adding the reference to this assembly once again. The Filter mechanism prompts you for a confirmation.
	Include. Includes the selected element. If you manually include a parent node, the filter mechanism includes the child elements below it, provided that no other rule exists. For example, if you include a class, it will include all its methods unless you specifically excluded a method.
	Exclude. Excludes the selected element. The child elements are also excluded unless they were included by another rule. By default, when you exclude a <i>class</i> , the filter mechanism applies the Exclude attribute to the class, but it allows the recording engine to record activity within the methods of the excluded class. When you exclude a <i>method</i> , however, the filter mechanism applies Totally Exclude, preventing the recording engine from recording any activity within the methods of the excluded class. Advanced users can modify these setting in the filter file.

	<p>Reset. Removes the manual inclusion or exclusion rule. In this case, the element may be impacted by other parent elements. The inclusion and exclusion rules have the following properties:</p> <ul style="list-style-type: none"> • The rules are hierarchical—if you add an include or exclude rule to a class, then the derived classes will follow the same rule unless otherwise specified. • A rule on a class only affects its public methods, derived classes, and inner classes. • A rule on a namespace affects all the classes and their public methods. • Note that adding or removing assemblies does not necessarily affect the classes that they contain—you can remove an assembly, yet its methods may be recorded due to the hierarchical nature of the filter. • As part of the filter design, several methods, such as .cctor() and Dispose(bool), do not follow the standard hierarchical rules. <div style="background-color: #e6f2e6; padding: 10px; margin-top: 10px;"> <p>Note: The resetting of a parent node does not override a manual inclusion or exclusion applied to a child node. For example, if you manually exclude a method, and then reset its class, which by default included all sub-nodes, your method will remain excluded.</p> </div> <p>Properties and events are view-only and cannot be included or excluded through the .NET Recording Filter pane. In addition, several system related elements are protected and may not be altered. For tips about including and excluding elements in the filter, see "Guidelines for Setting .NET Filters" on page 584.</p>
	<p>Navigate. Navigates to the previous or next tree node visited by the user.</p>
	<p>Show non-public items. By default, the filter tree shows only public classes and class members. By clicking this button, you instruct the tree view to display non-public items.</p> <p>If you include a class which contains non-public items, they will not be added to the filter automatically. You must explicitly include each non-public item to the filter.</p>
<p>Impact Log</p>	<p>Opens the Impact Log, which indicates what your last changes were and how they affected your filter. The user actions are listed in descending order, with the latest changes at the top. For each element, the log indicates how each manual inclusion or exclusion were affected. It also provides a link to the affected element in the filter pane hierarchy.</p>
<p>View Impact Log</p>	<p>Opens the Impact log for the selected filter. The Impact log shows which nodes in the tree were affected by recent actions.</p>

The following table shows the filter tree icons that represent the various elements:

Icon	Description	Icon	Description
	assembly		interface
	assembly that couldn't be loaded		method
	assembly that was partially loaded		static method
	class		namespace
	constructor		property
	static constructor		static property
	event		structure
	static event		

Options Dialog Box

The **Options** dialog box is comprised of VuGen application settings. These settings are common to all available protocols in VuGen.

General Options Tab

This pane enables you to configure general user interface options.

Task List

To access	VuGen > Tools > Options > General > Task List
------------------	--

User interface elements are described below:

UI Element	Description
------------	-------------

Comment Tags	<p>This pane enables you to add, delete or modify tags names that you can use to label comment tasks in your scripts.</p> <p>Tags. List of available tags.</p> <p>Name. Displays the name of the highlighted tags in the token list. This area enables you to modify, add or delete the current tag.</p> <p>Add. Enables you to add a tag to the token list.</p> <p>Edit. Enables you to modify the name of tag from the token list.</p> <p>Delete. Enables you to delete a tag from the token list.</p>
---------------------	---

Scripts and Solutions

To access	VuGen > Tools > Options > General > Scripts and Solutions
------------------	--

User interface elements are described below:

UI Element	Description
Settings	<p>Default project location</p> <p>Enables you to specify a path to your saved projects. Default location = C:\Users\<username>\Documents\SharpDevelop Projects</p> <p>Load previous solution on startup</p> <p>Enables you to automatically load the previous solution.</p> <p>By default this option is enabled.</p>
Start Page Settings	<p>Display Start Page on startup</p> <p>By default this option is enabled.</p> <p>Close Start Page after script loads</p> <p>By default this option is enabled.</p>

Git

Using these check boxes, you indicate which file types should be excluded from the Git repository.

To access	VuGen > Tools > Options > General > Message Dialogs
See also	"Working with Git" on page 120

User interface elements are described below:

UI Element	Description
Exclude replay data	Excludes any replay and runtime-data files associated with the script.
Exclude recorded snapshots	Excludes the snapshots generated during the script's recording.
Exclude MDRV logs	Excludes mdrv logs generated by VuGen during replay.
Exclude compilation files	Excludes files generated during the compilation stage.
Exclude script meta data	Excludes any meta data associated with the script.

ALM

To access	VuGen > Tools > Options > General > ALM
------------------	--

User interface elements are described below:

UI Element	Description
Connect to ALM in CAC mode	Connects to HPE Application Lifecycle Management using CAC (Common Access Card). This enables you to log in without providing a username and password.

Message Dialogs

Using these check boxes, you indicate which messages should be issued as popups and which should be hidden.

To access	VuGen > Tools > Options > General > Message Dialogs
------------------	--

User interface elements are described below:

UI Element	Description
Update Messages > Check for updates and display Update Available message	VuGen checks for updates at startup, and displays a message if an updated version is available.
Proxy Recording Tips Message	This tip instructs you how to proceed when recording an application that runs on another machine or device.
Web - HTTP/HTML Protocol JavaScript Message	This message appears when you create a Web - HTTP/HTML script, informing you that you can create your Vuser script in JavaScript.







Closing VuGen and Browser Message	This question appears when you close a TruClient script that is being edited in Interactive mode, warning you that the TruClient browser will be closed.
Community Search Privacy Warning	This warning message appears when you select the Search Community command from the context menu. It notifies you that the selected script text will be automatically passed into the search engine.
Paste runtime settings Warning	This warning appears when you paste runtime settings from the clipboard. It notifies you that the operation cannot be undone.
Action Reordered Warning	This notification informs you that rearranging actions within the solution tree will not affect the run logic.
Start Recording with IPv6 Warning	This warning notifies you that you cannot record an application with IPv6 with the current protocol.
Discard Correlation Warning	The Design Studio issues this message when you discard a correlation. It also explains how to restore it.

Community

To access	VuGen > Tools > Options > General > Community
See also	For details, see Community Search in the "Editor Pane" on page 54 .

User interface elements are described below:

UI Element	Description
Community Search Sites	

	<p>Adds a new search site to the list of Community search sites.</p> <ul style="list-style-type: none"> • Name: Enables you to specify a name of the search site that is displayed on the VuGen toolbar. • URL: Enables you to specify the URL of the search site. <div>  <p>Example:</p> <p><code>http://www.bing.com/search?q=%QUERY%</code></p> <p><code>http://www.google.com/search?q=%QUERY%</code></p> <p><code>http://www.google.de/search?q=%QUERY%</code> (localized google site, e.g. de for Germany)</p> <p><code>http://en.wikipedia.org/wiki/%QUERY%</code></p> </div>
	Enables you to edit the properties of the custom search site.
	Deletes the search site from the list of available sites.
	Moves the search site lower down on the list of available sites.
	Moves the search site higher up on the list of available sites.

Editor Options Tab

This pane enables you to configure the text editor options.

General

To access	Tools > Options > Editor > General
------------------	--

User interface elements are described below:

UI Element	Description
Font	

Text Font	Enables you to select the font.
Size	Enables you to select the font size.
General Options	
Word wrap	<p>Automatically wraps text to the next line.</p> <p>By default this option is disabled.</p>
Show Class/Function Browser	<p>Show or hide the Class/Function Browser. When enabled, you can navigate quickly to a specific class or function in your script by selecting it from the drop-down list in the browser.</p> <p>By default this option is enabled.</p>
Show line numbers	<p>Enable line numbering of script in the Editor.</p> <p>By default this option is enabled.</p>
Check for line ending inconsistencies	<p>Enable the editor to check for end of line inconsistencies in your script. If enabled, a utility appears in the script editor that enables you to normalize line endings based either on a Windows standard (CRLF) or a Linux standard (LF).</p> <div data-bbox="469 984 997 1197" data-label="Image"> </div> <p>This option is enabled by default.</p>
Enable working URL hypertext links in the Editor	<p>Enable URLs in scripts to function as hypertext links. Disabling this option may increase performance.</p> <p>By default this option is enabled.</p>

Markers and Rulers

To access	VuGen > Tools > Options > Editor > Markers and Rulers
------------------	--

User interface elements are described below:

UI Element	Description
Markers and Rulers	

Show spaces	<p>Enable markers that indicate where tabs exists.</p> <p>By default this option is disabled.</p>
Show tabs	<p>Enable markers that indicate where line ends.</p> <p>By default this option is disabled.</p>
Show end-of-line makers	<p>Enable markers that indicate where line ends.</p> <p>By default this option is disabled.</p>
Underline errors	<p>Enable underlining of errors.</p> <p>By default this option is enabled.</p>
Highlight matching brackets	<p>Enable highlighting of matching brackets.</p> <p>By default this option is enabled.</p>
Highlight symbols	<p>When this option is enabled, you can select a non-keyword in your script and the Editor will highlight all other occurrences in your script.</p> <pre> class MyClass { void foo() { int i; for (i = 0; i < 10; i++) { } int index; for(index = 0; index < 10; index++) while(true) { } while(true) { } } } </pre> <p>By default this option is enabled.</p>

Behavior

To access	VuGen > Tools > Options > Editor > Behavior
------------------	--

User interface elements are described below:

UI Element	Description
Tabs	
Indentation	Enables you to set the spacing for tab indentation. Default indentation is four spaces.
Convert tabs to spaces	Converts tabs to spaces. By default this option is disabled.
Use smart indentation	Automatically applies the indentation format from the previous line. By default this option is enabled.
Behavior	
Enable zoom with mouse wheel	Enables you to use the mouse wheel to zoom. By default this option is enabled.
Cut or Copy entire line when nothing is selected	Allows you to cut or copy an entire line without highlighting it in the editor, as long as the cursor is within that line. By default this option is enabled.
Enable Ctrl + Click for "Go to Definition"	Enables the Ctrl + Click shortcut for "Go To Definition". This lets you move the cursor to the definition of a function you control-click on in the editor. By default this option is enabled. To enable this option for an extra file containing custom functions, right-click on the file in the Solution Explorer and select "Add to Parsing List". Parsing list functionality applies only to C language functions and the Web HTTP/HTML protocol JavaScript functions.
Recording and Code Generation	
Use Compact mode for JavaScript scripts wherever possible	When using JavaScript as the coding language, use Compact mode. This mode reduces the number of lines in the script by selectively removing line breaks.

Code Color

To access	VuGen > Tools > Options > Editor > Highlighting
Important information	Highlighting options enable you to customize the color of script elements.

User interface elements are described below:

UI Element	Description
<Language Selection>	Drop-down list of script languages for which you can customize appearance including: <ul style="list-style-type: none">• C#• HTML• VuGen C• XML
<Element Selection>	List of code elements whose appearance you can customize.
Foreground color	Enables you to select a color from the pallet. The select color is applied to the foreground of the code element.
Background color	Enables you to select a color from the pallet. The select color is applied to background of the code element.

Code Completion

To access	VuGen > Tools > Options > Editor > Code Completion
------------------	---

User interface elements are described below:

UI Element	Description
Enable code completion features	Code completion features are enabled. For details on code completion, see "Editor Pane" on page 54 . By default this option is enabled.

Enable syntax tooltip	<p>Enables tooltips that display function arguments. Each argument is highlighted as you are defining it, moving to the subsequent argument when the delimiter is entered.</p> <div data-bbox="527 346 560 388"> </div> <p>Example:</p> <pre>int a ,b, c; a = my_add(1,2); b = my_add(2,4); c = my_add(5,</pre> <div data-bbox="857 625 1208 726"> <pre>int my_add(int a, int b)</pre> </div> <p>By default this option is enabled.</p>
Show tooltips when mouse pointer stops over an identifier	<p>When this option is enabled, a description of the code elements is displayed when the mouse hovers over the identifier. This option is disabled when Show tooltips in debug mode only is enabled.</p> <p>By default this option is enabled.</p>
Show tooltips in debug mode only	<p>Tooltips are displayed in debug mode only.</p> <p>By default this option is disabled.</p>
Include in the code completion list	
ANSI C keywords	<p>Enables you to include ANSI C keywords in code completion list.</p> <p>By default this option is enabled.</p>
LoadRunner API Steps	<p>Enables you to include LoadRunner API steps in the code completion list.</p> <p>By default this option is enabled.</p>
LoadRunner API Constants	<p>Enables you to include LoadRunner API constants in the code completion list.</p> <p>By default this option is disabled.</p>
User-defined functions	<p>Enables you to include user-defined functions in the code completion list.</p> <p>By default this option is enabled.</p>

Function parameters, local and global variables	<p>Enables you to include function parameters, local, and global variables in the code completion list.</p> <p>By default this option is enabled.</p>
--	---

Folding

To access	VuGen > Tools > Options > Editor> Folding
------------------	--

User interface elements are described below:

UI Element	Description
Enable folding features	<p>This option enables expanding and collapsing of script sections.</p> <p>By default this option is enabled.</p>
Enable steps folding	<p>This option enables expanding and collapsing of script steps.</p> <p>By default this option is disabled.</p>
When step length is more than [] characters	<p>Enables you define the number of characters in a step before implementing folding.</p> <p>By default this option is disabled.</p>
When step consists of more than [] lines	<p>Enables you to define the number of lines to a step before implementing folding.</p> <p>By default this option is disabled.</p>

Scripting Options Tab

This pane enables you to configure options related to recording, replaying and debugging scripts.

Recording

To access	Tools > Options > Scripting > Recording
------------------	---

User interface elements are described below:


UI Element	Description
Enable Recording Floating Toolbar transparency mode	<p>Display a transparent floating recording toolbar. When you click or hover on the toolbar it becomes opaque.</p> <p>By default this option is disabled.</p>

Enable 'Cancel Recording' button	<p>Enable the Cancel Recording button on the floating recording toolbar.</p> <div> <p>Note: When the Cancel Recording button is enabled, there may be a delay when you start recording into an existing script. This delay occurs while VuGen makes a copy of the existing script.</p> </div> <p>By default, this option is enabled.</p>
Open Start Recording Dialog Box after new script is created	<p>Automatically open the Start Recording Dialog Box after a new script is created.</p> <p>By default this option is disabled.</p>
Automatically close transactions	<p>Enable this function if you want VuGen to insert an end transaction step for open transactions before recording a subsequent action.</p> <p>By default this option is disabled.</p>
Check the Internet connection before recording	<p>Enable this function if you want VuGen to check for an Internet connection before recording starts.</p> <p>Note: The internet connection is checked by pinging google.com. If the Google server is not available (for example no external internet, Google is blocked, and so on) this could indicate a false-positive. In such cases, disable this check box.</p>
Check Edge recording permission before recording	<p>Enable this option if you want VuGen to check, before you start recording a script, if Microsoft Edge can record activity on intranet sites.</p>
Check Edge cache before recording	<p>Enable this option if you want VuGen, before you start recording a script, to check the Microsoft Edge cache: if the cache is not empty, some of the business process steps may not be recorded.</p>
Generate Recording Summary report	<p>Enables the generation of a Recording Summary report.</p> <p>To automatically display the report after generating the code, select Display the Recording Summary report when recording is finished.</p> <p>Note: The Recording Summary report is available for Web HTTP/HTML scripts only.</p>

Replay

To access	VuGen > Tools > Options > Scripting > Replay
------------------	---

User interface elements are described below:

UI Element	Description
Layout	<p>Do not switch the layout during Replay. By default, when you replay a script, VuGen automatically switches the UI layout to the Debug layout. This option enables you to maintain your selected layout during replay.</p> <p>By default this option is disabled.</p>
Animated Run	<p>Animated Run</p> <p>Runs the script in animated mode, highlighting the line in the script that is currently running. In non-animated mode, VuGen runs the script, but does not indicate the line being executed.</p> <p>By default this option is enabled.</p> <p>Animated Run Delay</p> <p>You can set a delay of the highlighting in the animated run, allowing you to better view the effects of each step. You set the delay in milliseconds.</p> <p>The default delay is 1.</p> <p>Animate Functions in the Action section only</p> <p>Animates the content of the Action sections only, not the init or end sections.</p> <p>By default this option is enabled.</p>
Results	<p>Enable result of replay summary to be saved to a named folder after each script run</p> <p>When this option is enabled the dialog box prompts you to name a results file before running a script in VuGen. When not enabled, VuGen automatically names the directory 'result1'. Subsequent script runs will automatically overwrite previous results files unless you specify a different name.</p> <div data-bbox="378 1436 1412 1520">  Note: Results are stored in a subdirectory of the script. </div> <p>By default this option is disabled.</p>

During Replay	<ul style="list-style-type: none"> • Show runtime view during replay Enables the runtime viewer • Auto Arrange Window Select this option to arrange the two viewers side by side. This option is disabled by default. • Collect replay statistics Select this option to enable the collection of replay-time statistics. The data collected is displayed in the Replay Summary report. This option is enabled by default. For details, see "Replay Summary Pane" on page 108. • Display NV Insights Report Select this option to enable the "NV Insights Report" on page 312. You open the NV Insights report from the Replay Summary page after you run a script. This option is enabled by default when the NV for Load Generator and VuGen component is installed on the VuGen machine. <div> <p>Note: This option is available only if the required Network Virtualization component is installed. For details on how to install the required Network Virtualization component, see the Network Virtualization for LoadRunner & Performance Center Help.</p> </div>
After replay	<p>After replay show Instructs VuGen how to proceed after the replay:</p> <ul style="list-style-type: none"> • Script. Show script in the Editor. • Replay summary. Go directly to the Replay Summary window in the Editor. (Default)

Script Management

To access	VuGen > Tools > Options > Scripting > Script Management
------------------	--

User interface elements are described below:

UI Element	Description
List of file types, by extension, that can be edited in the Editor	Enables you to modify the list of valid file extensions that can be edited in the Editor.

Comparison

To access	VuGen > Tools > Options > Scripting > Comparison
------------------	---

User interface elements are described below:

UI Element	Description
Path to comparing tool	<p>The comparison tool to be used when comparing two scripts.</p> <p>VuGen is installed with a default comparison tool (WinMerge). To use a different comparison tool, browse to the location of the tool on your local machine.</p>
Command line arguments - comparison tool	<p>A list of the command line arguments to use with your comparison tool.</p> <p>The mandatory default arguments %1 and %2 should not be modified.</p>
Path to merge tool	<p>The merge tool to use for combining files, for example, when "Working with Git" on page 120 repositories.</p> <p>VuGen is installed with a default merge tool (WinMerge). To use a different merge tool, browse to the location of the tool on your local machine.</p>
Command line arguments - merge tool	<p>A list of the command line arguments to use with your merge tool.</p> <p>The mandatory default arguments /e /u /dl %title1 /dr %title2 %1 %2 should not be modified.</p>

Step Navigator

To access	VuGen > Tools > Options > Scripting > Step Navigator
------------------	---

User interface elements are described below:

UI Element	Description
Enable Editor highlighting	This option enables you to highlight filtered steps in your script.
Background color	Enables you to select a background color to apply to the filtered steps in your script.
Border color	Enables you to select a border color to apply to the filtered steps in your script.

Thumbnails

To access	VuGen > Tools > Options > Scripting > Thumbnails
------------------	---

User interface elements are described below:


UI Element	Description
------------	-------------

Enable Thumbnail Explorer	Enables slide view of generated thumbnails in the Thumbnail Explorer. Double clicking a thumbnail sets the cursor at the associated step in the script.
Highlight the thumbnail associated with a step	Sync the display of the Thumbnail Explorer while scrolling through steps in the Editor.
Show important thumbnails by default	VuGen displays thumbnails directly related to the business process and filters out less important thumbnails by default.
Enable Automatic Creation	Enables the automatic creation of thumbnails during the application's idle time.
Cache thumbnails to script folder	Optimize VuGen's performance by saving rendered thumbnails to a cache file. Thumbnails are loaded from the cache file after the initial generation.

Output Pane

To access	VuGen > Tools > Options > Scripting > Output Pane
------------------	--


User interface elements are described below:

UI Element	Description
Format	Word wrap Enables word wrapping in the Output pane.
Font	Text Font Enables you to select a font for the text that appears in the Output pane. Size Enables you to select a font size for the text that appears in the Output pane.
Use color coding to display text in the Output pane	Enables color coding of the text that appears in the Output pane. Color coding may slow down system response while a very large output log is displayed (rendered) in the Output pane or as you scroll through the output log. <div>  Note: Restart VuGen for changes to this option to take effect. </div>

Java

To access	VuGen > Tools > Options > Scripting > Java
------------------	---

User interface elements are described below:

UI Element	Description
Eclipse IDE Location	Browse Enables you to set the location of the Eclipse program, eclipse.exe .  Note: For details on supported Eclipse versions, see the System Requirements .

Citrix

To access	VuGen > Tools > Options > Scripting > Citrix
------------------	---

User interface elements are described below:

UI Element	Description
Show client during replay	Shows the Citrix client during script replay.
Show Bitmap Selection popup	Issues a popup message when inserting a Get Text or Sync Bitmap function before selecting a bitmap or text in the snapshot.

Correlation

To access	VuGen > Tools > Options > Scripting > Correlation
------------------	--

User interface elements are described below:

UI Element	Description
Enable correlation from replay snapshots	Enables the Create Correlation context menu item in the Replay snapshot's Response Body pane. This option instructs VuGen to format the data so that it will be open for correlations. When disabled, VuGen only generates raw data which cannot be altered.

Snapshots

To access	VuGen > Tools > Options > Scripting > Snapshots
------------------	--

User interface elements are described below:

UI Element	Description
------------	-------------

Enable snapshot viewing	Enables the viewing of snapshots when you click on a step in the editor or step navigator. This option allows you to improve VuGen's performance when working with very large scripts.
Enable enhanced XML view	<p>Enables the following XML viewer visual features:</p> <ul style="list-style-type: none"> • XML tree • coloring <p>Clear this option to reduce the amount of memory consumed by the XML view.</p>
Enable snapshot caching	<p>Allows snapshots to be cached.</p> <p>Clear this option where you are working with large snapshots and are running out of memory.</p>
Do not load text snapshots larger than	Text-based snapshots are not loaded if they are larger than the specified size.
Do not load binary snapshots larger than	Binary [hexadecimal] based snapshots are not loaded if they are larger than the specified size.
Do not load XML snapshots larger than	XML-based snapshots are not loaded if they are larger than the specified size.
Do not highlight XML snapshots larger than	XML-based snapshots are not highlighted if they are larger than the specified size.
Enable HTTP Page View option in the Snapshot pane	Enables the display of HTTP protocol snapshots in the Page View. Disabled by default because it may affect VuGen performance. It is recommend to use the HTTP Data view instead.

Parameters

To access	VuGen > Tools > Options > Scripting > Parameters
------------------	---

User interface elements are described below:

UI Element	Description
-------------------	--------------------

Left parameter delimiter	<p>Enables you to specify a left parameter delimiter.</p> <p>The following characters are valid: !, #, \$, %, &, (,), [,], {, }, , ~, ` , <, >, ?</p> <div> <p>Note: If you change the left parameter delimiter, the specified delimiter will be applied to new Vuser scripts only, not to existing scripts.</p> </div>
Right parameter delimiter	<p>Enables you to specify a right parameter delimiter.</p> <p>The following characters are valid: !, #, \$, %, &, (,), [,], {, }, , ~, ` , <, >, ?</p> <div> <p>Note: If you change the right parameter delimiter, the specified delimiter will be applied to new Vuser scripts only, not to existing scripts.</p> </div>
Parameter background color	Enables you to specify the background color for parameters in a Vuser script.
Parameter border color	Enables you to specify the border color for parameters in a Vuser script.
Restore Delimiter Defaults	Resets both the left and right parameter delimiters to their default values.

Parser

To access	VuGen > Tools > Options > Scripting > Parser
------------------	---

User interface elements are described below:

UI Element	Description
Enable C Language Parser	<p>Disabling the C language parser may improve application performance when you are working with very large scripts. However, the following features will be disabled:</p> <ul style="list-style-type: none"> • Editing step arguments in the Editor • Statement completion • Snapshots • Tasks • Thumbnails • Additional step-related functionality <p>This option is enabled by default.</p>



Search and Replace Dialog Boxes

These dialog boxes enable you to find and replace text strings in Vuser scripts and solutions.

Search Dialog Box

To access	<ul style="list-style-type: none"> • Search > Quick Find • Search > Find in Files
Important information	<p>VuGen's find mechanism does not perform a cyclic search and will stop when it reaches the end of the last file in the specified scope.</p> <p>Tip: To perform another search, click Find Next at the end of the search.</p>

User interface elements are described below (unlabeled elements are shown in angle brackets). Some of the UI elements in the table below only appear in the **Quick Find** dialog box, while others only appear in the **Find in Files** dialog box:


UI Element	Description
<Search-type drop-down>	<p>Enables you to specify the type of search to perform: Quick Find or Find in Files.</p> <div>  <p>Note: The Search dialog box user interface changes depending on the selection.</p> </div>
Find text	The text or expression to search for.
	Shows the Regular Expression Builder. The Regular Expression Builder enables you to build a regular expression in the Find text box.
Regular Expression	Indicates that the Find text string is a regular expression.
Scope	The scope of the search: Selection, Current Action, Current Script, or Entire Solution. You move the slide bar to indicate your selection.
Include in search	Indicates the entities to include in the search: Code editor, snapshots, and logs.
Directory	For a "Find in Files" search, the folder that contains the files that will be searched.
Options	Additional search options, such as case and whole word matching, or the direction of the search (for a Quick Find search).
Find Next	Finds the next occurrence of the text or regular expression in the Find text box.

Find All	<p>Finds all occurrences of the text or regular expression that appears in the Find text box. The results appear in the Search Results pane. In the Search Results pane, you can:</p> <ul style="list-style-type: none"> • Display the results by file, or as a flat list. • Right-click an entry and select Locate to show the corresponding text in the Vuser script or wherever it is located. • Right-click an entry and select Copy to copy the selected search result to the clipboard. • Right-click and select Copy All to copy all the search results to the clipboard.
-----------------	--

Replace Dialog Box

To access	Search > Quick Replace
------------------	----------------------------------

User interface elements are described below:


UI Element	Description
Find text	Specify the text to search for.
	Shows the Regular Expression Builder. The Regular Expression Builder enables you to build a regular expression in the Find text box.
Regular Expression	Select Regular Expression to indicate that the Find text string is a regular expression.
Replace with	The text that will replace the Find text .
Scope	The scope of the search: Selection, Current Action, Current Script, or Entire Solution. You move the slide bar to indicate your selection.
Options	Enables you to specify search options, such as case and whole word matching, or the direction of the search.
Find Next	Finds the next occurrence of the text or regular expression in the Find text box.
Replace	Replaces the selected text with the text in the Replace with box.
Replace All	Replaces all found occurrences of the text or regular expression in the Find text box with the text in the Replace with box.

Business Process Report Dialog Box

This dialog box enables you to create a Business Process report.

To access	Tools > Business Process Report
Relevant tasks	"Create a Business Process Report" on page 140


User interface elements are described below:

UI Element	Description
Title	The title that will appear on the report.
Author	Your name, as it will appear in the report.
Comment	Additional comments that you want to appear in the report.
Location	The location where you want the report to be saved. Default value: Script folder.
 More	Expands the Business Process Report dialog box to display more options.
Table of contents	Includes a table of contents in the report. If you disable this option, a table of contents will not appear in the report. Default value: Enabled.
Recording summary	A summary of the recording session, as it appears in the Replay Summary pane, is included in the report. Default value: Enabled.
Transactions and Rendezvous lists	The report includes a list of all of the transactions and rendezvous points that were defined in the script. Default value: Enabled.
Parameters list	A list of all the parameters that were defined in the script. This list corresponds to the parameters listed in the Parameter List dialog box Design > Parameters > Parameter List . Default value: Enabled.
Thumbnails	The report includes an actual snapshot of each recorded step, adjacent to the step name and description. Default value: Enabled.
Step descriptions	The report includes a short description of each step. Default value: Enabled.

UI Element	Description
Script name	The .usr file name of the script.
Output format	Creates the report in the selected format. The following formats are available: <ul style="list-style-type: none">• Microsoft Word• Adobe PDF• HTML
Document template	The path and file name of the template to use for the report. The default template is usually stored in the product's dat folder. To change the report template, click the browse button and specify a new template with a .docx extension. If you want to create a new template, we recommend that you use an existing template as a basis for the new one. This will make sure that the required bookmarks and styles are maintained within the new template.

Replay Summary Pane


This pane provides summarized replay results and links to script replay details.

To access	Use one of the following: <ul style="list-style-type: none">• Solution Explorer > Right click Replay Runs > and then select Open Replay Summary• Click the View summary link in the Output Pane
Important information	To enable transaction breakdown data, select Tools > Options > Scripting > Replay > Collect replay statistics .  Note: Enabling the Collect replay statistics option will affect replay performance.
See Also	<ul style="list-style-type: none">• "Output Pane" on page 77• "Replay a Vuser Script" on page 278


UI Element (unlabeled elements are shown in angle brackets)	Description
Results Dashboard	<p>Displays basic script information including:</p> <ul style="list-style-type: none"> • Script name • Replay Status: Displays a replay status of the script as either Script Passed or Script Failed. • Elapsed time: Total time passed during script replay. • Started at. Starting time of the script replay. • Ended at. Ending time of the script replay. • Think time. Total duration of Think time¹ passed during script replay. • Wasted time. Total duration of Wasted time² passed during script replay. • Save HAR file. For Web HTTP/HTML scripts only. Saves the traffic information to a HAR file. • Export to PDF. Saves the current replay summary to a PDF file. • Open Network Virtualization Insights Report. Opens the .
< Script Performance >	<p>A list of the script's actions and transactions with basic information. Click on an action or transaction to show or hide its details below the table (only available if you enable the collection of replay statistics) .</p> <ul style="list-style-type: none"> • Name: The name of the action or transaction. • Duration: When the scope is a single iteration, the time displayed is the duration of the transaction. When the scope is an average, the time displayed is the average duration of all iterations. • Duration Trend: A Sparkline representation of values over all iterations (for multiple iterations only). • Status: The number of iterations with a status of Passed/Total iterations.

¹Span of time inserted into a script to simulate a user's pausing before moving on to the next step in a business process.

²Time spent on activities whose purpose is to support test analysis, but would never be performed by a browser user.


UI Element (unlabeled elements are shown in angle brackets)	Description
Connection statistics Details	<p>A collection of statistics per action or transaction, as per the selected scope (the script average, a specific action, or a specific iteration).</p> <div> Note: These statistics are only available for Web - HTTP/HTML protocol scripts.</div> <ul style="list-style-type: none">• Count: The number (or average number for multiple iterations) of connections to the server per domain.• Hit Count: The number (or average number for multiple iterations) of files requested from the server.• Hit Count %: The amount of each hits per item (or average for multiple iterations) as a percentage of the total hits to all items. Expand the node to view all of the items.• Hit Count % Trend: A Sparkline representation of the hit count values over all iterations.• Size: The size (or average size for multiple iterations) of the data returned from the server per domain.• Size %: The percentage (or average percentage for multiple iterations) of data size returned from the server per domain, of the total returned data.• Size % Trend: A Sparkline representation of values over all iterations.

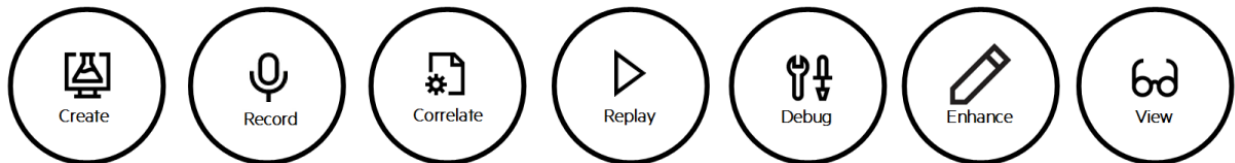
UI Element (unlabeled elements are shown in angle brackets)	Description
<p>Responses per content-type</p> <p>Details</p>	<p>Content-type statistics per transaction or action for the current scope (the script average, a specific action, or a specific iteration).</p> <div data-bbox="764 443 1412 567"> <p>Note: These statistics are only available for Web HTTP/HTML protocol scripts.</p> </div> <ul style="list-style-type: none"> • Responses per content-type: A list of the content type returned from the server, for example, an image, a JavaScript, a CSS (expand to view list). • Count: The number (or average number for multiple iterations) of connections per content type. • Count %: The percentage (or average percentage for multiple iterations) of connections to the server content type from the total number of connections. • Count % Trend: A Sparkline representation of the count values over all iterations. • Size: The size (or average size for multiple iterations) of the data returned from the server per content type. • Size %: The percentage (or average percentage for multiple iterations) of data returned from the server for each content type from the total returned data. • Size % Trend: A Sparkline representation of values of the data size per content type over all iterations.

UI Element (unlabeled elements are shown in angle brackets)	Description										
Responses per HTTP status Details	<p>HTTP Status statistics per action or transaction for the current scope (the script average, a specific action, or a specific iteration).</p> <div data-bbox="764 443 1412 567"> <p>Note: These statistics are only available for Web - HTTP/HTML protocol scripts.</p> </div> <ul style="list-style-type: none"> • Responses per HTTP Status: List of the HTTP status codes returned . • Count: The number (or average number for multiple iterations) of connections per HTTP status code. • Count %: The percentage (or average percentage for multiple iterations) of connections per HTTP status code from the whole action or transaction. • Size % Trend: A Sparkline representation of values over all iterations. 										
Replay Statistics Summary <table border="1" data-bbox="212 1045 698 1299"> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Total Hits</td><td>18.25</td></tr> <tr> <td>Total Throughput (bytes)</td><td>33727.75</td></tr> <tr> <td>New Connections</td><td>1.5</td></tr> <tr> <td>Connection Shutdowns</td><td>1.5</td></tr> </tbody> </table>	Name	Value	Total Hits	18.25	Total Throughput (bytes)	33727.75	New Connections	1.5	Connection Shutdowns	1.5	<p>A list of script statistics and their values after replay, per scope (the script average, a specific action, or a specific iteration).</p>
Name	Value										
Total Hits	18.25										
Total Throughput (bytes)	33727.75										
New Connections	1.5										
Connection Shutdowns	1.5										
<Replay Button> 	<p>Enables you to replay your script from the Replay Summary pane.</p>										
<Modify Runtime Settings>	<p>Enables you to access Runtime Settings for your active script.</p>										
<Modify Content Check Rules>	<p>VuGen's ContentCheck mechanism enables you to detect all types of errors sent by the web server.</p> <p>For details, see Internet Protocol > ContentCheck view in the runtime settings.</p>										

VuGen Workflow

The **VuGen Workflow** section provides you with an understanding of the steps you need to follow in order to create an effective load testing script.

 Select an image to learn more.



Creating or Opening Vuser Scripts


This section describes how to create new Vuser scripts, and open existing ones.

What do you want to do?

- [Learn more about creating scripts](#)
- [Create a new script](#)
- [Open a script from ALM](#)
- [Create a multiple protocol script](#)
- [Save my script to an archive file](#)

Creating Vuser Scripts - Overview

Creating a Vuser script includes the steps shown below. This topic provides an overview of the first step, creating a Vuser script.

 Select an image to learn more.



The first step in developing a Vuser script is to create a blank script. For details on how to create a blank Vuser script, see ["Create and Open Vuser Scripts" on the next page](#). The contents and structure of the blank Vuser script vary slightly based on the protocol of the script. Therefore, before you create a blank Vuser script, you must know the protocol to use for the script. After you create a blank Vuser script, you are ready to perform the next step in the script creation workflow - recording user actions into the script. For details, see ["Recording" on page 131](#).

When you create a Vuser script, VuGen creates a series of configuration files, data files, and source code files that comprise the Vuser script. These files contain Vuser runtime and setup information. For details on the files that comprise a Vuser script, see ["Script Directory Files" on page 127](#).

VuGen enables you to:

- Create or open a script from a template. For task details, see ["Create and Open Vuser Script Templates" on page 130](#).
- Open or work with a .zip script. You can unzip or work with a script in .zip format. For task details, see ["How to Work with .zip Files" on page 129](#).
- Open a script stored in Application Lifecycle Management. For more information, see ["Working with Application Lifecycle Management" on the next page](#).
- Use *Application Lifecycle Management* (ALM) to store and retrieve Vuser scripts, scenarios, and analysis results. You can store scripts in an ALM project and organize the scripts into unique groups. For more information, see ["Managing Scripts Using ALM - Overview" on the next page](#).

Create and Open Vuser Scripts

To:	Do this:
Create a new Vuser script	<ol style="list-style-type: none"> 1. Open VuGen and select File > New Script and Solution. 2. In the Create a New Script dialog box, select Single Protocol or Multiple Protocols from the Category list. 3. Select a protocol from the Protocols list. 4. In the Script Name box, enter a name for the script. Note: Do not name scripts <i>init</i>, <i>run</i> or <i>end</i> because these names are used internally by VuGen. 5. Click Create to create the Vuser script. <p>For user interface details, see "Create a New Script Dialog Box" on page 128.</p> <p>After you create a new Vuser script, you can record user activity into the script. For details, see "Record a Vuser Script" on page 135.</p>
Create or open a script from a template	For task details, see "Create and Open Vuser Script Templates" on page 130 .
Open an existing script stored on the local machine or network drive	Select File > Open > Script/Solution .
Open or work with a .zip script	For task details, see "How to Work with .zip Files" on page 129 .

To:	Do this:
Open a script stored in Application Lifecycle Management	You can store scripts on ALM and modify them in VuGen. For details, see " Working with Application Lifecycle Management " below.

Compare Scripts Side-by-Side

Vuser scripts can be compared and displayed side by side using the comparison tool.

1. Right-click the primary script in **Solution Explorer** and select **Set as first comparing object**.
2. Right-click the secondary script in the **Solution Explorer** and select one of the following:
 - **Compare**
 - **Compare with external object** to compare an asset to a file outside of the solution. This sets the highlighted asset as the primary asset and opens Windows Explorer to enable you to select the secondary asset.

Note: You can change the comparison tool from **Options > Scripting > Comparison**. For more information, see "[Scripting Options Tab](#)" on page 96.

Working with Application Lifecycle Management

The **Working with Application Lifecycle Management** section describes how to manage your Vuser scripts by integrating with Application Life Cycle Management.

Managing Scripts Using ALM - Overview

VuGen works together with *Application Lifecycle Management* (ALM). ALM provides efficient functionality for storing and retrieving Vuser scripts, scenarios, and analysis results. You can store scripts in an ALM project and organize the scripts into unique groups.

In order for VuGen to access an ALM project, you must connect VuGen to the Web server on which the ALM project is located. You can connect to either a local or remote Web server.

For details on working with ALM, see the *HPE Application Lifecycle Management User Guide*.

Connect to ALM

To store and retrieve scripts from ALM, you need to connect to an ALM project. You can connect or disconnect from an ALM project at any time during the testing process.

You can connect to one version of ALM from VuGen and a different version from your browser. For more information, see the **Important Information** section in "[HPE ALM Connection Dialog Box \[VuGen\]](#)" on page 119.

Connect to a project in ALM

1. Determine the type of authentication required for the ALM server: User name/password or CAC (Common Access Card). For CAC mode, enable CAC authentication in VuGen's **General** options. For details, see ["General Options Tab" on page 86](#).
2. Select **Version Control > ALM > ALM Connection**. The HPE ALM Connection dialog box opens.
3. In the **Step 1: Connect to server** section, enter a user name and password (not relevant for CAC authentication) and click **Connect**. VuGen connects to the ALM server.
To disconnect from ALM, click **Disconnect**.
4. In the **Step 2: Login to project** section, enter the domain and project details, and then click **Login**. VuGen logs in to the specified project.
To log out of the project, click **Logout**.
5. Click **Close** to close the HPE ALM Connection dialog box.

Note: If you authenticated through CAC mode and disconnected from the ALM server, you need to restart VuGen before reconnecting in CAC mode.

ALM Version Control - Overview

VuGen supports version control features in Vuser scripts saved in ALM projects that use version control.

The version control features change the process of opening and saving a script. Scripts with version control are either in a state of checked-in or checked-out. When you are working with a script in a checked-out state, any changes you make will not be saved on the ALM server until you check in the script. If you save the script from within VuGen, a temporary file is saved on your machine that protects your changes in case your computer crashes.

If you are working with a script in a checked-in state, the script is read-only and you cannot make any changes until you check out the script.

If a particular script is being saved to ALM for the first time, and the project uses version control, the script automatically starts in a checked-out state.

Work with Scripts in ALM Projects

The following steps describe the workflow of how to work with Vuser scripts that are saved in an ALM project.

Note: To work with scripts in ALM projects with version control, see ["Work with Version-Controlled Scripts in ALM Projects" on the next page](#).

1. Connect to ALM

Open a connection to the ALM server and project that contains the script. For task details, see

["Connect to ALM" on page 115.](#)

2. Open the script

Select **File > Open > Script/Solution**. In the Open VuGen Script or Solution dialog box, select the script to open and then click **Open**.

3. Save the script

Select **File > Save Script**. If the script is in a project that uses version control and is not checked out, the script is saved as a temporary file on your local machine.

Work with Version-Controlled Scripts in ALM Projects

Relevant for: Scripts in ALM projects that support version control AND have the Performance Center addition installed.

If these two conditions are not met, see ["Work with Scripts in ALM Projects" on the previous page.](#)

The following steps describe how to work with scripts saved in ALM projects for which version control is enabled.

1. Connect to ALM.

Open a connection to the ALM server and project that contains the script. For task details, see ["Connect to ALM" on page 115.](#)

2. Open the script.

Select **File > Open > Script/Solution**. In the Open VuGen Script or Solution dialog box, select the script to open and then click **Open**.

3. Check in/out the script.

If the ALM project has version control, each script is always defined as being either checked-in or checked-out. For more details, see ["ALM Version Control - Overview" on the previous page.](#) To check in and check out scripts, select **Version Control > ALM > Check In/Check Out**.

Note: If the ALM project has version control, the file is locked when it is checked out.

If the ALM project is not version controlled, the file is not locked when checked out of the project.

4. Cancel a check out. (Optional)

If you checked out a script and do not want to save the changes, you can return the status of the script to checked-in without saving by selecting **Version Control > ALM > Undo Check Out**.

5. Save the script.

Select **File > Save Script**. If the script is in a project that uses version control and is not checked out, the script is saved as a temporary file on your local machine.

Save VuGen Vuser Scripts to ALM Projects

The following steps describe how to save a Vuser script to an ALM project.

1. Open or create the Vuser script

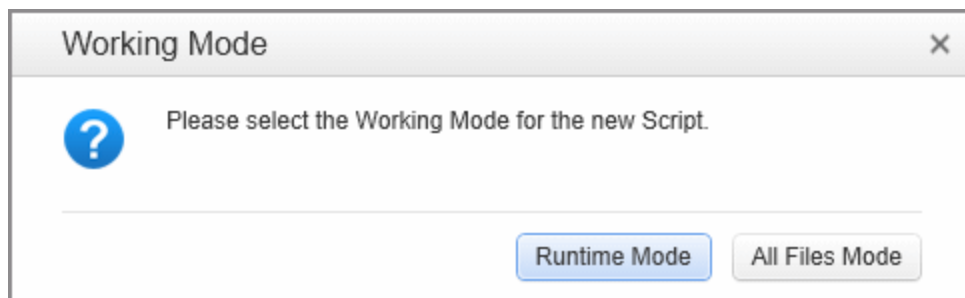
Create or open the desired script in VuGen.

2. Connect to ALM

Open a connection to the ALM server and project in which you want to store the script. For task details, see ["Connect to ALM" on page 115](#).

3. Save the script to ALM

- Select **File > Save Script as**. The Save Script As dialog box opens.
- Click **ALM Test Plan**, and then specify the name and location for the script.
- Click **Save**. After a short time, the Working Mode dialog box opens.



- Select one of the following options:

Runtime Mode. Copies only the files needed to replay the script. This option does not copy recording snapshot files and other unnecessary files. This results in a shorter transfer time.

All Files Mode. Copies all of the files associated with this script. This results in a longer transfer time.

Compare Previous Versions of a Script

If your Vuser script is saved in an ALM project that uses version control, you can compare previous versions of the script. The following steps describe how to do this.

1. Connect to ALM

Open a connection to the ALM server and project that contains the script that you want to view or modify. For task details, see ["Connect to ALM" on page 115](#).

2. Open the script

Select **File > Open > Script/Solution**. In the Open VuGen Script or Solution dialog box, select the

script to open and then click **Open**.

3. Compare previous versions of the script


- Select **Version Control > ALM > Version History**. The Version History dialog box opens.
- Select two previous versions of the script and then click **Compare Versions**. WDiff opens and displays the two versions of the script.


HPE ALM Connection Dialog Box [VuGen]

This dialog box enables you to connect to an ALM project from within VuGen.

To access	Version Control > ALM > ALM Connection
Important information	<ul style="list-style-type: none"> You can connect to one version of ALM from VuGen and a different version of ALM from your browser. You can connect to different versions of ALM only if one of the versions is ALM 11.00 or higher. <div style="border: 1px solid #ccc; background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: Before you connect to results stored on ALM through the this dialog box, it is recommended that you first connect to the ALM server through your browser. This automatically downloads the ALM client files to your computer.</p> </div> <ul style="list-style-type: none"> You can configure more advanced settings, such as a proxy setting, by using the Webgate Customization tool (webgatecustomization.exe) and then sign into ALM using the HPE ALM Connection Dialog Box. The Webgate Customization tool can be found on your ALM server at the following address: <code>http://<ALM Server>/qcbn/Apps/</code>.

User interface elements are described below:

UI Element	Description
Step 1: Connect to Server	<ul style="list-style-type: none"> Server URL. The URL of the server on which ALM is installed. User name. Your ALM project user name (not relevant for CAC authentication). Password. Your ALM project password (not relevant for CAC authentication).  Connect. Connects to the server specified in the Server URL box. Disconnect. Disconnects from the current ALM server.

Step 2: Login to Project	<ul style="list-style-type: none"> • Domain. The domain that contains the ALM project. Only those domains containing projects to which you have permission to connect to are displayed. • Project. Enter the ALM project name or select a project from the list. The list includes only those projects to which you have permission to connect. •  . Logs into the ALM project. • Logout. Logs out of the current ALM project.
Restore connection on startup	Automatically reconnect to the ALM server the next time you start VuGen, using the same credentials.

Working with Git

The following sections describe how to work with a Git repository to store and retrieve your scripts.

Managing Scripts with Git - Overview

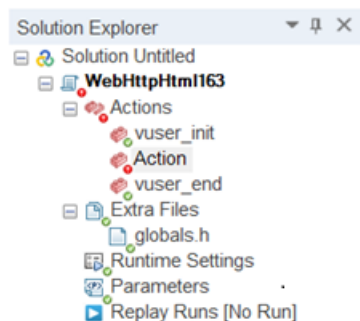
VuGen integrates with Git Hub, allowing you to upload scripts from a Git repository and perform common actions such as Pull, Push and Track.

VuGen's **Version Control** menu allows you to import a script from a Git repository. For details, see ["Import from Remote Repository Dialog Box" on page 126](#).

You can also take a local script and add it to a Git repository, directly from the Solution Explorer. Using the right-click menu, you can perform typical ["Git Operations" on page 122](#). You can also view history and manage your changes.

To limit the files added to your repository, VuGen uses a **.gitignore** file, which can be edited. For details, see ["Configure Ignore List Dialog Box" on page 124](#).

For scripts stored in a Git repository, the Solution Explorer pane indicates which files changed since the last commit.



For details about working with a Git repository script, see ["Work with Scripts in Git" on the next page](#).

Work with Scripts in Git

The following steps describe the workflow of how to work with Vuser scripts and a Git repository. For an overview of the GitHub integration, see ["Managing Scripts with Git - Overview" on the previous page](#).

1. Open or create a Git repository script in VuGen

- **For a local script not part of a Git Repository:**

Create or open a local script in VuGen. Click the parent level of the script in the Solution Explorer and select **Git > Create Local Git Repository**. The Track Script dialog box opens. Accept the default gitignore content, or click **Configure Ignore List** to edit the list or use an external file. For details, see ["Configure Ignore List Dialog Box" on page 124](#). After you edit the ignorelist, click **Track**.

- **For a script stored in a Git Repository:** Download the script from the Git repository.

Select **Version Control > Git Options > Import from Remote Repository**. Specify the repository information and click **Import**. The script opens in the VuGen editor. For details, see ["Import from Remote Repository Dialog Box" on page 126](#).

2. Edit the script

Edit or record the script as you would any other script.

3. Perform a Pull operation (optional)

Click the parent level of the script in the Solution Explorer and expand the **Git** entry from the right-click menu. Select **Pull** to retrieve the updated files from the Git repository. For details, see ["Git Operations" on the next page](#). If there are conflicts between the files, VuGen opens the ["Resolve Conflicts Dialog Box" on page 125](#).

4. Commit the changes

Save the file in VuGen. Click the parent level of the script in the Solution Explorer and select **Git > Commit...** from the right-click menu. In the Commit Changes dialog box, specify the change's details and click **Commit**.

5. Push the changes to the Git repository

Click the parent level of the script in the Solution Explorer and expand the **Git** entry from the right-click menu. Select **Push** to send the changes to the Git repository. For details, see ["Git Operations" on the next page](#). If there are conflicts between the files, VuGen opens the ["Resolve Conflicts Dialog Box" on page 125](#).

6. Track or untrack scripts (optional)

To disable tracking for a script, click its parent level in the Solution Explorer and select **Git > Untrack Script**. To re-enable tracking, select **Git > Activate Script Tracking**.

For additional options, such as undoing changes or viewing history, see ["Git Operations" on the next page](#).

Troubleshooting

If you are unable to perform push, pull, or import operations:

- Make sure you have an active Internet connection.
- Make sure that a proxy server is not blocking the connection. Configure the Git machine to allow

access via your proxy server. For details, see the Git documentation.

If you push to a remote repository and then change the connection to use a second remote repository, and push the commits to that second repository, the push fails. To push to the second repository:

1. Change the connection to point to the second repository.
2. Perform a new commit.
3. Push to the new repository.

Git Operations



VuGen allows you to perform Git operations from the script's context (right-click) menu.

To access	<ol style="list-style-type: none"> 1. Open a script stored in a Git repository. <ul style="list-style-type: none"> • For a new script that is not part of a Git repository, select Git > Create local Git repository from right click menu. • For a script that already exists in GitHub, import it, as described in "Import from Remote Repository Dialog Box" on page 126. 2. In the Solution Explorer pane, click the parent level of the script and expand the Git item in the right-click menu.
Important information	<ul style="list-style-type: none"> • If the script is on a private repository and you need to Pull or Push, you must provide credentials. • Make sure your machine is configured to integrate with Git and that you set the relevant proxy settings, if applicable.
Relevant tasks	<ul style="list-style-type: none"> • "Work with Scripts in Git" on the previous page
See also	<ul style="list-style-type: none"> • "Import from Remote Repository Dialog Box" on page 126 • "Configure Ignore List Dialog Box" on page 124

The Git context menu operations are described below in alphabetical order:

UI Element	Description
Activate Script Tracking	<p>(Available only for untracked scripts) Adds the current script to the staging area for tracking. To ignore specific files from the tracking, use the Configure Ignore List option.</p> <div style="border-left: 2px solid green; padding-left: 10px; margin-top: 10px;"> <p>Note: If you rename a tracked file outside of VuGen, the Solution Explorer will display its new name.</p> </div>

UI Element	Description
Check for Modifications...	<p>Displays the changes to the script files since your last commit. Double-click on an entry to open the changes in a Diff tool.</p> <p>Expand the Internal files and User files sections to see which files were modified.</p>
Commit...	<p>Opens the Commit Changes dialog box, allowing you to:</p> <ul style="list-style-type: none"> Type a description of the change in the Message area. See which files will be committed. Expand the Internal files and User files sections to see the file list. Edit the signature for the change by clicking Edit... Add the script files to the staging area and commit the changes with the specified message.
Configure Ignore List...	<p>Allows you to modify the .gitignore content or import the contents of a previously saved .gitignore file.</p> <p>Place each file that you want to ignore on a separate line. Use an asterisk as a wildcard. For complete syntax information, see the Git documentation.</p>
Connect to Git...	<p>Opens the Connect to Git dialog box, allowing you to specify the connection information:</p> <ul style="list-style-type: none"> Remote Repository URL. The URL of the repository to use for Push, Pull, and Commit operations. User name, Password. The credentials to use for connecting to the repository. They are saved internally to ensure persistent connectivity.
Create Local Git Repository...	<p>Opens the Track Script dialog box. Click Track to add the local script to a Git repository.</p> <p>To indicate which script files to exclude from the repository, click Configure ignored list to open the "Configure Ignore List Dialog Box" on the next page.</p>
Pull	<p>Pulls the changes from the repository to your local script files.</p>
Push...	<p>Opens the Push Changes dialog box, when changes are detected. If you are not connected to Git, VuGen opens the Connect to Git dialog box, prompting you for credentials.</p> <p>In the Changes section, you select the change that you want to push to the repository.</p> <p>Expand the Internal files and User files sections to see which files will be included in the push operation.</p>

UI Element	Description
Refresh	Recalculates and updates the tree icons in VuGen's Solution Explorer pane, with the latest actions and files.
Resolve Conflicts...	<p>Opens the Resolve Conflicts dialog box, to help you resolve conflicts between local files and those on the repository.</p> <p>Select the check box adjacent to each file whose conflict you want to mark as resolved.</p> <p>Double-click on a conflicted file to open the merge tool.</p> <p>For additional details, see "Resolve Conflicts Dialog Box" on the next page.</p>
Reset HEAD	<p>Sets the files in the working directory and staged snapshot of the current branch, HEAD, to their content at your last commit.</p> <div>  Caution: This operation will discard all of your uncommitted changes. </div>
Revert...	<p>Allows you to revert to a specific commit.</p> <p>In the Revert a Commit dialog box, select a commit from the list and click Revert. The revert command will only undo the changes made in the selected commit. All other commits are unaffected.</p> <div>  Note: When reverting a commit that has more than one parent, the revert will use the first parent. </div>
Show log...	<p>Opens the Log dialog box, listing the changes and their associated files.</p> <p>In the Files section, expand the Internal files and User files headers to see the files that were modified for the selected change.</p>
Untrack Script	(Available for a tracked script only) Removes the script and all its files from the list of tracked files.

Configure Ignore List Dialog Box

This dialog box allows you to set the gitignore content for the current script.

To access	Right-click the script's parent node in the Solution Explorer, and select Git > Configure Ignore List from the context menu (for scripts already in the repository).
Relevant tasks	"Work with Scripts in Git" on page 121

See also	"Git Operations" on page 122
-----------------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<ignore file list>	An editable area for listing the files to ignore. Use an asterisk (*) as a wildcard. Place each entry on a separate line.
Import	Allows you to import an existing .gitignore file to the script.
Select	Opens the Ignore File Templates dialog box, allowing you to select a predefined template for an ignore list. You can select one of the following templates: <ul style="list-style-type: none"> • Default template. Ignores all temporary files and those generated automatically by VuGen. • Replay Files template. Ignores all files that are not required for replay.
Wrap text	Wraps the text in the ignore file list to the next line.

Resolve Conflicts Dialog Box

This dialog box allows you to import scripts or resources from a Git repository.

To access	<ol style="list-style-type: none"> 1. Open a script stored in a Git repository. <ul style="list-style-type: none"> • For a new script that is not part of a Git repository, select Git > Create local Git repository from right click menu. • For a script that already exists in GitHub, import it, as described in "Import from Remote Repository Dialog Box" on the next page. 2. In the Solution Explorer pane, click the parent level of the script and expand the Git item in the right-click menu and select Resolve Conflicts. <p>This dialog box will also open automatically if you attempt to perform a Pull operation for files that are in conflict.</p>
Relevant tasks	"Work with Scripts in Git" on page 121
Important Information	You set a merge tool from the "Options Dialog Box" on page 86 (Tools > Options > Scripting > Comparison). Double click an entry to open it in the merge tool.

User interface elements are described below:

UI Element	Description
<file list>	A list of User and Internal files that are in conflict. Double-click on a file to open a merge tool that will allow you to resolve the conflict. For details about setting up a merge tool, see above.
Mark as Resolved	Marks the files whose check boxes are selected, as resolved. To mark all files as resolved, choose Select all files .

Import from Remote Repository Dialog Box

This dialog box allows you to import scripts or resources from a Git repository.

To access	Version Control > Git Options > Import from Remote Repository
Relevant tasks	"Work with Scripts in Git" on page 121
See also	"Git Operations" on page 122

User interface elements are described below:

UI Element	Description
Remote repository URL	The URL of the repository from which you want to import the resource.
Repository name	The name of the repository to import.
Local Path	The local path on which to save the Git resources.

Multiple Protocol Scripts

When you record a single protocol, VuGen records only the specified protocol. When you record in multi-protocol mode, VuGen records the actions in several protocols. To see which Vuser types are supported for multi-protocol recording, click the **Multiple Protocols** node in the Create a New Script dialog box. For details, see ["Create a New Script Dialog Box" on page 128](#).

Another variation between Vuser types is multiple-action support. Most protocols support more than one action section. Currently, the following protocols support multi-actions: Oracle NCA, Web - HTTP/HTML, RTE, and C Vusers.

For most Vuser types, you create a new Vuser script each time you record—you cannot record into an existing script. However, when recording a Java, Web - HTTP/HTML, Oracle NCA, or RTE Vuser script, you can also record within an existing script.

Since VuGen supports a large variety of protocols, some of the recording steps that follow apply only to specific protocols.

For protocol-specific information, see:

Java	"Java Record Replay Protocol" on page 516
MQTT	"MQTT Protocol" on page 570

In SOA (Service Oriented Architecture) systems, it is essential that you test the stability of your applications and services before deployment. VuGen allows you to create basic Web Service scripts.

















Unified Functional Testing (UFT), HPE's functional testing tool, contains additional features that help you create a comprehensive testing solution for your SOA environment. For more information, contact an HPE representative.

Script Directory Files

While you create a Vuser script, VuGen creates a series of configuration files, data files, and source code files that comprise the Vuser script. These files contain Vuser runtime and setup information. VuGen saves these files together with the script in the script folder.

To access the files in the script folder:

Right-click the script name in the **Solution Explorer** and select **Open Script Folder**.


 data	5/6/2013 4:27 PM	File folder	
 Action	5/6/2013 4:28 PM	C Source	25 KB
 AsyncCallbacks	5/6/2013 4:27 PM	C Source	9 KB
 Breakpoints	5/6/2013 4:28 PM	XML Document	1 KB
 custom_body_variables	3/13/2001 7:46 PM	Text Document	1 KB
 default.cfg	5/6/2013 4:27 PM	CFG File	2 KB
 default.usp	5/6/2013 4:28 PM	USP File	3 KB
 globals	5/6/2013 4:28 PM	C/C++ Header	1 KB
 lrw_custom_body	7/14/2010 6:16 PM	C/C++ Header	1 KB
 ReplaySummaryReport	5/6/2013 4:27 PM	XML Document	2 KB
 ThumbnailsCache.tmp	5/6/2013 4:30 PM	TMP File	2,591 KB
 UserTasks	5/6/2013 4:28 PM	XML Document	1 KB
 vuser_end	1/5/2012 11:52 AM	C Source	1 KB
 vuser_init	1/5/2012 11:52 AM	C Source	1 KB
 WebHttpHtml1	5/6/2013 12:20 PM	SQL Server Compac...	128 KB
 WebHttpHtml1	5/6/2013 4:28 PM	Virtual User Test	1 KB

See also:

- ["Files Generated During Recording" on page 227](#)



Create a New Script Dialog Box

This dialog box enables you to create a new Vuser script.

To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> • File > New Script and Solution • File > Add > New Script • Click the  button in VuGen.
Relevant tasks	<ul style="list-style-type: none"> • "Creating or Opening Vuser Scripts" on page 113 • "Record a Vuser Script" on page 135 • "How to Record a Script with TruClient - Mobile Web" on page 567

User interface elements are described below:

UI Element	Description
Category / Protocol	<p>The protocols to display in the Protocol pane and the type of script to create (for all except for User Templates):</p> <ul style="list-style-type: none"> • User Templates. Displays any user-defined templates you have created. Available only if at least one user-defined template exists. For details, see "Create and Open Vuser Script Templates" on page 130. • Single Protocol. Shows all of the protocols. When you click OK, it creates a single protocol script using the selected protocol. • Multiple Protocols. Shows all of the protocols that can be included in a multi-protocol script. Select the check boxes adjacent to the protocols that you want to include. • Mobile and IoT. Shows all of the protocols for creating a script for a mobile application or Internet of Things (IoT) application. <ul style="list-style-type: none"> Mobile. For details, see "Select a Recording Method for Mobile Applications" on page 563. IoT. For details, see "MQTT Protocol" on page 570. • Popular. Shows a list of the most commonly used protocols. • Recent. Shows a list of the most recently used protocols.
Filter	<p>Enables you to filter the protocol list by entering text. For example, if you type "Java" into the Filter box, the Protocol list will display only those protocols that include the word Java.</p>

Script Name	<p>Enables you to specify the name of your script.</p> <p>If you create a single protocol script, the default name is <protocol name>x where x represents the numerical sequence of the script created. For example, the name of the third script created for the Web- HTTP/HTML protocol would be WebHttpHtml3.</p> <p>If you create a multi-protocol script, the default name is <protocol name_multi>x where protocol name is the first protocol you selected from the list and x represents the numerical sequence of the script created.</p>
Location	<p>Enables you specify the file location of your script. You can use the browse button to navigate to a location on your file system.</p> <p>Tools > General > Projects and Solutions enables you to specify a default location.</p>
Solution Name	<p>This option is displayed only when a solution is not open in the Solution Explorer. You can specify a name for the solution. If you leave it blank, the default name is 'Untitled'.</p>
Create folder for solution	<p>Enables you to create a folder for your solution.</p>
Solution Target	<p>Displays the file path of the solution.</p>
	<p>Displays the protocols in list view.</p>
	<p>Displays the protocols in icon view.</p>

How to Work with .zip Files

Compressing script files into a .zip file conserves disk space and makes it easier to copy or transfer your scripts from one machine to another.

Import from a Zip File

To open a script stored in a .zip file, select **File > Manage Zip Files > Import from Zip File**. After you select a .zip file, VuGen prompts you for a location in which to store the unzipped files.

Note: If you import a script from an archive file that was archived only with runtime files, you will not be able to regenerate the script to its original recorded state. For details, see ["Regenerate a Vuser Script" on page 220](#).

Export to a Zip File

To save the entire script folder as a .zip file, select **File > Manage Zip Files > Export to Zip File**.

You can indicate whether to save all the files or only the runtime files.

Note: If you export the script with the runtime files only, the user of the imported script will not be able to regenerate the script to its original recorded state.

Zip and Email

To create a .zip file and send it as an email attachment, select **File > Manage Zip Files > Zip and Email**. When you click **OK** in the **Zip and Email** dialog box, VuGen compresses the file according to your settings and opens an email compose form with the .zip file as an attachment.

Edit Script in Zip File

To work from a .zip file, while not expanding or saving the script files, select **File > Manage Zip Files > Edit Script in Zip File**. When you modify the script and save it, the changes are stored directly in the .zip file.

Create and Open Vuser Script Templates

Create a Vuser Script Template

1. Open a script in VuGen.
2. Select **File > User-Defined Templates > Export to Template**.
3. Enter a name and location for the template.
4. Click **OK** to create the template.

Create a Vuser Script From a Template

Select **File > New Script and Solution > VuGen > User Templates** and select the template file (only available after you create at least one template).

Rename a Vuser Script Template

1. Select **File > User-Defined Templates > Manage from Explorer**
2. In the Explorer dialog box:
 - a. Rename the content file (.zip).
 - b. Rename the description file (.xpt).
3. Using a text editor, modify the following tags in the .xpt file:

- a. **Name tag:** `<Name>NewName </Name>`
- b. **File tag src property:** `<File name="template_temp.zip" src="NewName.zip" binary="True" />`

Template Guidelines

- If you configure a script for a specific protocol and then save the script as a template, further scripts based on that template will work only with that same protocol.
- After you create a template, you cannot edit it directly in VuGen. To make any changes, open the template and save it again with another name or overwrite the existing template.
- If you regenerate an original script from a template, you will lose all of your manual changes.

Vuser Script Templates

The User-Defined Template enables you to save a script with a specific configuration as a template. You can then use this template as a basis for creating future scripts.

User-defined templates support the following files and data:

- Runtime settings
- Parameters
- Extra files
- Actions
- Snapshots

Generic settings, such as Recording options and General options, are not supported as they are not relevant to a specific script.

Template Guidelines

- If you configure a script for a specific protocol and then save the script as a template, further scripts based on that template will work only with that same protocol.
- After you create a template, you cannot edit it directly in VuGen. To make any changes, open the template and save it again with another name or overwrite the existing template.
- If you regenerate an original script from a template, you will lose all of your manual changes.

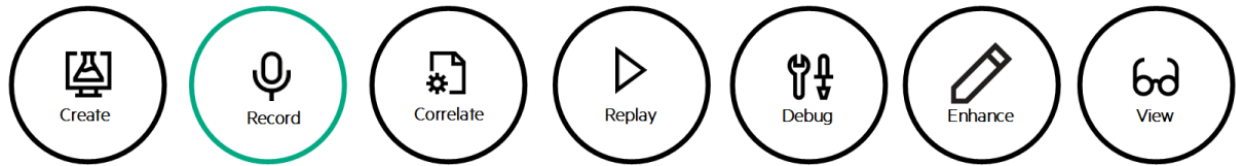
Recording

The **Recording** section describes script sections, script recording, working with templates, and other recording tools.

Recording - Overview

Creating a Vuser script includes the steps shown below. This topic provides an overview of the second step, recording a Vuser script.

Select an image to learn more.



After you create an empty Vuser script, you are ready to use VuGen to record typical user-actions into the script. While you record the script, VuGen's floating Recording toolbar gives you access to the main recording functionality, such as pausing and stopping the recording, and inserting transactions and rendezvous points. For details on how to record a Vuser script, see ["Record a Vuser Script" on page 135](#).

Each Vuser script contains at least three sections: *vuser_init*, one or more action sections, and *vuser_end*. When you run multiple iterations of a Vuser script, only the *Actions* sections of the script are repeated—the *vuser_init* and *vuser_end* sections are not repeated. Before you record, and during recording, you can select the section of the script into which VuGen will insert the recorded functions. For details on the script sections, see ["Vuser Script Sections" below](#).

Before you start recording, make sure that the recording options are set correctly for the script. For more information about the recording options, see ["Recording Options" on page 141](#).

When you have finished recording the user actions, VuGen generates the Vuser script and performs various other post-recording operations. You can replay the script to make sure that it functions correctly. For details, see ["Replaying" on page 277](#).

To resolve situations where you cannot install VuGen on the client machine, VuGen allows you to record scripts using a LoadRunner proxy. Proxy recording may be required with certain Linux machines, Mac OS machines, and mobile devices. For details, see ["Recording via a Proxy - Overview" on page 214](#)

After you successfully record a Vuser script, you can replay the script. For details, see ["Replay Overview" on page 277](#).

Vuser Script Sections

Each Vuser script contains at least the following sections:

Script Section	Used when recording...	Is executed when...
<i>vuser_init</i>	a login to a server	the Vuser is initialized (loaded)
<i>Actions</i>	client activity	the Vuser is in Running status
<i>vuser_end</i>	a logoff procedure	the Vuser finishes or is stopped

Before and during recording, you can select the section of the script into which VuGen will insert the recorded functions.

When you run multiple iterations of a Vuser script, only the *Actions* sections of the script are repeated—the *vuser_init* and *vuser_end* sections are not repeated. For more information on the iteration settings, see the General > Run Logic view in the Runtime settings.

VuGen Script Editor

You use the VuGen script editor to display and edit the contents of each of the script sections. You can display the contents of only a single section at a time. To display a section in the script editor, double-click the name of the section in the Solution Explorer.

Java Classes

When working with Vuser scripts that use Java classes, you place all your code in the Actions class. The Actions class contains the following methods: `init`, `action`, and `end`. These methods correspond to the sections of scripts developed using other protocols—you insert initialization routines into the `init` method, client actions into the `action` method, and log off procedures in the `end` method.

```
public class Actions{  
    public int init() {  
        return 0;}  
    public int action() {  
        return 0;}  
    public int end() {  
        return 0;}  
}
```

Note: Transaction Breakdown for Oracle DB is not available for actions recorded in the `vuser_init` section.

For more details, see ["Java Vuser Protocol" on page 536](#).

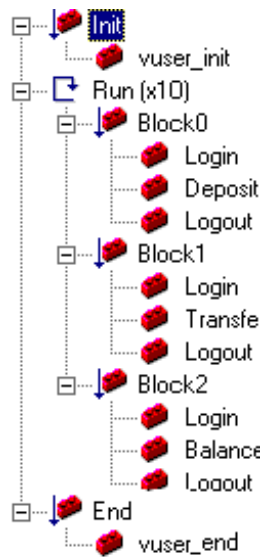
Script Section Structure Example

Every Vuser script contains three sections: `vuser_init`, *Run (Actions)*, and `vuser_end`. You can instruct a Vuser to repeat the *Run* section when you run the script. Each repetition is known as an *iteration*.

The `vuser_init` and `vuser_end` sections of a Vuser script are not repeated when you run multiple iterations.

When you run scripts with multiple actions, you can indicate how to execute the actions, and how the Vuser executes them:

In the following example, *Block0* performs a deposit, *Block1* performs a transfer, and *Block2* submits a balance request. The *Login* and *Logout* actions are common to the three blocks.



Sequence. You can set the order of actions within your script. You can also indicate whether to perform actions sequentially or randomly.

Iterations. In addition to setting the number of iterations for the entire *Run* section, you can set iterations for individual actions or action blocks. This is useful, for example, in emulating a commercial site where you perform many queries to locate a product, but only one purchase.

Weighting. For action blocks running their actions randomly, you can set the *weight* or percentage of each action within a block.

In most cases, the name of the header file corresponds to the prefix of the protocol. For example, Database functions that begin with an **lrd** prefix, are listed in the **lrd.h** file.

Header Files

Header files commonly contain forward declarations of classes, subroutines, variables, and other identifiers. In most cases, the name of the header file corresponds to the prefix of the protocol. For example, Database functions that begin with an **lrd** prefix, are listed in the **lrd.h** file.

The following table lists the header files associated with the most commonly used protocols:

Protocol	File
Ajax (Click & Script)	web_ajax.h
Citrix	ctrxfuncs.h
COM/DCOM	lrc.h
Database	lrd.h
FTP	mic_ftp.h

Protocol	File
General C function	lrun.h
IMAP	mic_imap.h
LDAP	mic_mldap.h
MAPI	mic_mapi.h
Oracle NCA	orafuncs.h
POP3	mic_pop3.h
RDP	lrrdp.h
SAP GUI	as_sapgui.h
Siebel	lrsiebel.h
SMTP	mic_smtp.h
Terminal Emulator	lrrte.h
Web (HTML\HTTP)	as_web.h
Web (Click & Script)	web_api.h
Web Services	wsoap.h
Windows Sockets	lrs.h

Record a Vuser Script

1. Create a new script or open an existing script. For details, see ["Creating or Opening Vuser Scripts" on page 113](#).
2. (Recommended) Modify the Windows DEP settings.
 - a. Open **Start > Control Panel > System**.
 - b. In the Advanced tab, click **Performance settings**.
 - c. In the Performance Options Data Execution Prevention tab, select the first option, **DEP for essential services only**.
If you cannot change this option, click **Add**. Browse to the client program, for example IEXPLORE.EXE.
 - d. If neither of these options is possible, try to disable DEP completely.
 - i. Open a command prompt.
 - ii. Run the following command: **bcdedit.exe /set {current} nx AlwaysOff**

- iii. Reboot the machine.
 - iv. Verify that the settings took effect by running the following at the command line:
BCDEdit /enum
 - v. Verify that **nx** is **AlwaysOff**.
3. (Optional) Configure the recording options.


The recording options affect the way a Vuser script is recorded and how it is generated after the recording. For details, see ["Recording Options" on page 141](#).



Tip:

- **Web HTTP/HTML scripts:** To generate a Recording Summary report after recording is finished see ["Enable the Recording Summary report" on the next page](#).
- If the business process you want to record contains asynchronous push communication, select **Recording Options > HTTP > Advanced > Use streaming mode when recording with the LoadRunner Proxy**.
- If you are not able to successfully record a script with VuGen, select **Recording Options > HTTP > Advanced Node** and check the **Use LR Proxy to record a local application** option. Then rerecord your business process.


4. Start the recording session.

To start recording, click the **Record** button  on the VuGen toolbar, make the relevant selections in the Start Recording dialog box, and click **Start Recording**. VuGen's floating toolbar appears, VuGen opens your application and begins recording your actions.


- For user interface details, see ["Start Recording Dialog Box" on page 221](#).
- For details on the script sections into which you can record, see ["Vuser Script Sections" on page 132](#).

5. Perform business processes on your application.

Perform the desired business processes that you wish to record. The floating toolbar allows you to insert transactions, rendezvous points, and comments. You can also use the floating toolbar to specify into which section of the script to record. For user interface details, see ["Floating Recording Toolbar" on page 225](#).

Click the **Stop** button  on the floating toolbar when you are finished recording.



Note: To cancel the recording session, click the **Cancel Recording** button  on the floating toolbar. When you cancel a recording, VuGen removes all of the code that was added to the script during the current recording session, thereby restoring the script to its status before the current recording session. For details on enabling/disabling the **Cancel Recording** button, see ["Scripting Options Tab" on page 96](#).

Recording Summary Report

When you record your application, unwanted information is often included in the script, such as Google statistics, advertisements, and third party information.

VuGen's Recording Summary report enables you to make decisions about which parts of the recording to include in your generated script, and which to filter out.

The Recording Summary provides general information about your recording session, as well as detailed information about hosts, headers, and content types. The report also enables you to manage traffic filtering.

Note: This feature is available for Web HTTP/HTML scripts only.

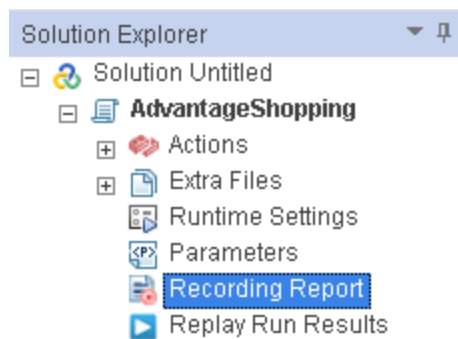
Enable the Recording Summary report

To configure VuGen to automatically generate a Recording Summary report at the end of a recording session:

1. In VuGen, select **Tools > Options**.
2. On the **Scripting** tab, under **Recording**, select **Generate Recording Summary report**.
3. To open the report after it is generated, select **Display the Recording Summary report when recording is finished**.

Access the Recording Summary report

Assuming **Generate Recording Summary report** is enabled (see ["Enable the Recording Summary report" above](#)), when you finish recording your script the Recording Summary report appears in the script tree under the **Recording Report**.



If your options are set to display the report when recording is finished, the report opens in the Script Editor pane.

Recording Summary report overview

The Recording Summary report displays a dashboard which contains top-level details about your recording session.



Tip: Click the button in the upper-right corner to expand the report.

1 correlations were found

[Open Design Studio](#)

[Open Recording Options](#)



The report displays the following:

Page	Description
Dashboard	<p>Displays:</p> <ul style="list-style-type: none"> General recording information: <ul style="list-style-type: none"> Total size: Details about the recorded script including throughput, number of requests, an number of connections recorded. <p>Note: When using WinINet mode, an unexpected number of connections may appear.</p> <ul style="list-style-type: none"> Date and duration of the recording The client application or browser Number of correlations found during the recording session The number of headers found during the recording session Distribution of content types and response codes Host information, by domain
Hosts	<p>Displays of the hosts that were recorded during the recording session, grouped by domain.</p> <p>The following information is provided about each host:</p> <ul style="list-style-type: none"> Host's IP address SSL version and SSL Cipher used for the connection HTTP version Number of requests made Total size of data received from the host during the recording session
Headers	<p>Displays information about the HTTP headers created during the recording.</p> <p>Headers that are included (selected) are added in a web_add_header function.</p> <p>Note: This page is informational only, and cannot be edited.</p>

Page	Description
Content Types	Displays information about the content types recorded: <ul style="list-style-type: none"> how many times they appeared during the recording session total size and download time of matching data received

Filter recorded script using the Recording Summary report

- To drill down into the report details:

Detail	Click....
Correlations	Open Design Studio button
Headers	x Headers were recorded or More details link
Hosts	More details link
Content Types	More details link

- Use the search bar on the **Hosts**, **Headers**, or **Content-Type** pages to find a particular detail.
- On the **Hosts/Headers** pages, select the **Included/Excluded** buttons to view a list of hosts/headers in your script. The **Included/Excluded** buttons display the number of hosts/headers included or excluded. When both buttons are selected (default), all of the hosts/headers are listed.

On the Hosts page, you can select whole domains, or particular hosts within domains, to include/exclude in the script.

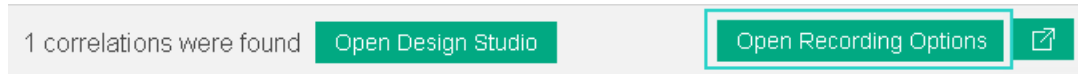
Note: The Headers page is informational only, and cannot be edited.

- On the **Content Types** page, select the **Non-Resource/Resource** buttons to view a list of non-resource and resource content types found in the script. The **Non-Resource/Resource** buttons display the number of content types that are non-resources or resources. When both buttons are selected (default), all of the content types found in the script are listed.

Use the togglers next to each content type to define it as a non-resource or resource.

NON-RESOURCE	
<input type="radio"/>	Resource
<input type="radio"/>	Resource
<input checked="" type="radio"/>	Non-Resource
<input type="radio"/>	Resource

- When you filter the hosts or content types, the filter is added to the **Recording Options**. To manually edit the filter lists, click the **Open Recording Options** button at the top of the report.



For host and traffic filtering information, see ["Network > Mapping and Filtering Recording Options" on page 185](#).

For Resources/Non-resources filtering information, see ["Non-Resources Dialog Box" on page 181](#).

Regenerate the script

When you make changes in the report, red asterisks appear next to the changed entities and the **Regenerate** button appears at the bottom of the screen. Click to regenerate the script.



The changes are applied to the script.

Note: Manual enhancements to a recorded action step are lost when regenerating the script. For more details, see ["Regenerate a Vuser Script" on page 220](#).

Create a Business Process Report

At the final stage of script creation, you can create a report that describes your business process.

Types of information included in business process reports

You can edit or update the business process report template according to your requirements, as well as customize the contents of the report by indicating the type of information to include.

Item	Supported types
Script export formats	<ul style="list-style-type: none">• Microsoft Word• Acrobat PDF• HTML
Template types	<ul style="list-style-type: none">• Pre-designed template• Built-in VuGen template (Microsoft Word 2007 (.docx) format)

Item	Supported types
Protocols	<ul style="list-style-type: none">• Ajax (Click & Script)• TruClient• Citrix• Oracle NCA• Oracle - Web• RDP• SAP GUI• SAP - Web• Web - HTTP/HTML• Web Services

Create a business process report

Select **Tools > Business Process Report** and fill in the required data. For user interface details, see ["Business Process Report Dialog Box" on page 106](#).

Configure additional report options

Click the **More** button to modify options such as the table of contents, snapshots, and the document template. For user interface details, see ["Business Process Report Dialog Box" on page 106](#).

Recording Options

The **Recording Options** section describes the many different options that affect your Vuser script during the recording and generation stages of creating a script.

Citrix > Configuration Recording Options

Enables you to set the window properties and encryption settings for the Citrix client during the recording session.

To access	Record > Recording Options > Citrix > Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Encryption Level	The level of encryption for the ICA connection: Basic, 128 bit for login only, 40 bit, 56 bit, 128 bit , or Use Server Default to use the machine's default.
Window Size	The size of the client window. Default value: 800 x 600.

Citrix > Code Generation Recording Options

Enables you to configure the way VuGen captures information during recording.

To access	Record > Recording Options > Citrix > Code Generation
Important information	<ul style="list-style-type: none"> This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204. Text synchronization steps that you add manually during the recording are not affected by the above settings—they appear in the script even if you disable the above options. The synchronization options also work for regenerating a script. For example, if you originally recorded a script with Add text synchronization calls disabled, you can regenerate after recording to include text synchronization.

User interface elements are described below:

UI Element	Description
Use Citrix Agent input in Code Generation	<p>Use the Citrix Agent input to generate a more descriptive script with additional synchronization functions. Default value: enabled.</p> <ul style="list-style-type: none"> Automatically generate text synchronization calls. Adds text synchronization Sync on Text steps before each mouse click. Default value: disabled.

Citrix > Login Recording Options

Enables you to you set the connection and login information for the recording session.

To access	Record > Recording Options > Citrix > Login
------------------	---

Important information	<ul style="list-style-type: none"> The Login node is available only when creating a single protocol Citrix script. When creating a multi-protocol Citrix+Web script, the login information is retrieved from the Web interface. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204. If you do not provide login information, you are prompted for the information when the client locates the specified server.
------------------------------	--

User interface elements are described below:

UI Element	Description
Connection	<ul style="list-style-type: none"> Network Protocol. The preferred protocols are TCP/IP and TCP/IP+HTTP. Most Citrix Servers support TCP/IP, however some Citrix clients do not. Certain servers are configured by the administrators to allow only TCP/IP with specific HTTP headers. If you encounter a communication problem, select the TCP/IP+HTTP option. Server. The Citrix server name. To add a new server to the list, click Add, and enter the server name (and its port for TCP/IP + HTTP). <div> <p>Note: Multiple servers apply only when you specify a Published Application. If you are connecting to the desktop without a specific application, then list only one server.</p> </div> Published Application. The name of the Published Application as it is recognized on Citrix server. The drop-down menu contains a list of the available applications. If you do not specify a published application, VuGen uses the server's desktop. If you added or renamed a published application, close the Recording options and reopen them to view the new list. Additionally, you can also enter the name of a published application manually if you know it exists (useful in cases where the drop-down list is inaccurate). To change the name of the published application on the Citrix client, you must make the change on the Citrix Server machine. Select Manage Console > Application and create a new application or rename an existing one. <div> <p>Note: If you do not specify a published application, Citrix load balancing will not work. To use load balancing when accessing the server's desktop, register the desktop as a published application on the server machine, and select this name from the Published Application drop-down list.</p> </div>

Define connection parameters	Allows you to manually define the logon and connection details.
Logon Information	Specify the User Name , Password , and Domain of the Citrix user. Optionally, you can also specify the Client Name by which the Citrix server identifies the client.
Use ICA file for connection parameters	Specify an ICA file with the connection configuration information for the application. For details, see "Citrix > Login Recording Options" on page 142 .

ICA File Structure

Citrix ICA client files are text files that contain configuration information for the applications accessed through the Citrix client.



Tip: When you load an ICA file using the Recording Options, VuGen saves the file together with your script, eliminating the need to copy the ICA file to each load generator machine.

ICA Files must have an **.ica** extension and must conform to the following format:

```
[WFClient]
Version=
TcpBrowserAddress=

[ApplicationServers]
AppName1=

[AppName1]
Address=
InitialProgram=#
ClientAudio=
AudioBandwidthLimit=
Compress=
DesiredHRES=
DesiredVRES=
DesiredColor=
TransportDriver=
WinStationDriver=
Username=
Domain=
ClearPassword=
```

The following example shows a sample ICA file for using Microsoft Word on a remote machine through the Citrix client:

```
[WFCClient]
Version=2
TcpBrowserAddress=235.119.93.56

[ApplicationServers]
Word=

[Word]
Address=Word
InitialProgram=#Word
ClientAudio=On
AudioBandwidthLimit=2
Compress=On
DesiredHRES=800
DesiredVRES=600
DesiredColor=2
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
Username=test
Domain=user_lab
ClearPassword=test
```

For more information, see the Citrix website www.citrix.com.


Citrix > Recorder - Recording Options

Enables you to configure how window names are generated when the window title changes during recording and to configure whether to save snapshots of the screens.

To access	Record > Recording Options > Citrix > Recorder
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see " Protocol Compatibility Table " on page 204.

User interface elements are described below:

UI Element	Description
Save snapshots	Saves a snapshot of the Citrix client window for each script step, when relevant. We recommend that you enable this option to provide you with a better understanding of the recorded actions. Saving snapshots, however, uses more disk space and slows down the recording session.

Window name	<p>In some applications, the active window name changes while you are recording. If you try to replay the script as is, the Vuser uses the original window name and the replay may fail. You can specify a naming convention for the windows in which VuGen uses a common prefix or common suffix to identify the windows as follows:</p> <ul style="list-style-type: none"> • Use new window name as is. Set the window name as it appears in the window title. (default) • Use common prefix for new window names. Use the common string from the beginning of the window titles as a window name. • Use common suffix for new window names. Use the common string from the end of the window titles as a name. <p>Alternatively, you can modify the window names in the actual script after recording. In the Script view, locate the window name, and replace the beginning or end of the window name with the "*" wildcard notation.</p> <div>  Example: <code>ctrx_sync_on_window ("My Application*", ACTIVATE, ...CTRX_LAST);</code> </div>
--------------------	--


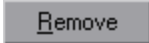
COM/DCOM > Filter Recording Options

Enables you to define which COM/DCOM objects to record.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> • Record > Recording Options > COM/DCOM > Filter • Replay > Recording Options > COM/DCOM > Filter
------------------	---

User interface elements are described below:

UI Element	Description
DCOM Profile	<p>Specify one of the following filter types:</p> <ul style="list-style-type: none"> • Default Filter. The filter to be used as the default when recording a COM Vuser script. • New Filter. A clean filter based on the default environment settings. Note that you must specify a name for this filter before you can record with its settings. <p>You can also save the current settings and delete a filter using the Save As and Delete buttons.</p>

<p>DCOM Listener Settings List</p>	<p>Displays a tree hierarchy of type libraries. You can expand the tree to show all of the available classes in the type library. You can expand the class tree to show all of the interfaces supported by that class.</p> <p>To exclude a type library, clear the check box next to the library name. This excludes all of its classes in that type library. By expanding the tree, you can exclude individual classes or interfaces by clearing the check box next to the item.</p> <p>An interface can be implemented differently by various classes. When you exclude an interface that is implemented by other classes that have not been excluded, a dialog box opens asking you if you also want to exclude the interface in all classes that implement it this interface.</p> <p>Note that when you clear the check box adjacent to an interface, it is equivalent to selecting it in the Excluded Interfaces dialog box.</p> <ul style="list-style-type: none"> • Environment. The environments to record: ADO objects, RDS Objects, and Remote Objects. Clear the objects you do not want to record. • Type Libraries. A type library .tlb or .dll file, that represents the COM object to record. All COM objects have a type library that represents them. You can select a type library from the Registry, Microsoft Transaction Server, or file system. <p>Type Libraries. In the lower section of the dialog box, VuGen displays the following information for each type library.</p> <ul style="list-style-type: none"> • TypLib. The name of the type library (tlb file). • Path. The path of the type library. • Guid. The Global Unique Identifier of the type library.
<p></p>	<p>Adds another COM type library.</p> <ul style="list-style-type: none"> • Browse Registry. Displays a list of type libraries found in the registry of the local computer. Select the check box next to the desired library or libraries and click OK. • Browse file system. Allows you to select type libraries from your local file system. • Browse MTS. add a component from a Microsoft Transaction Server. The MTS Components dialog box prompts you to enter the name of the MTS server. <p>Type the name of the MTS server and click Connect. Remember that to record MTS components you need an MTS client installed on your machine.</p> <p>Select one or more packages of MTS components from the list of available packages and click Add. Once the package appears in the list of Type Libraries, you can select specific components from the package.</p>
<p></p>	<p>Removes a COM type library.</p>

<div>Exclude...</div>	<p>Excludes interfaces in the filter through the Excluded Interfaces dialog box. In this dialog box, the checked interface listings are the ones that are excluded. You can also add interfaces that are not listed. Click Add Interface... in the Excluded Interfaces dialog box and enter the GUID number (interface ID) and name of the interface. You can copy the GUID from the interfaces.h file created by VuGen and listed in the selection tree in the left-hand column of the VuGen screen. Use the Add Interface... feature to exclude interfaces that are called needlessly by the script, but are not listed anywhere in the filter.</p> <p>An interface can be implemented differently by various classes. When you exclude an interface that is implemented by other classes that have not been excluded, VuGen displays the a warning. If you check Don't ask me again and close the dialog box, then the status of all instances of the interface in all other classes will be changed automatically for this filter, whenever you change the status of the interface in one object. Click Yes to all to change the status of all instances of this interface for all other classes, click No to all to leave the status of all other instances unchanged. Click Next Instance to view the next class that uses this interface.</p>
-----------------------	--

COM/DCOM > Options Recording Options

Enables you to set additional options for your COM recording session, relating to the handling of objects, generation of logs, and VARIANT definitions.

The DCOM scripting options apply to all programming languages. These settings let you configure the scripting options for DCOM methods and interface handling.

To access	Record > Recording Options > COM/DCOM > Options
------------------	---

User interface elements are described below:

UI Element	Description
ADO Recordset filtering	Condense multiple recordset operations into a single-line fetch statement (enabled by default).
Declare Temporary VARIANTs as Globals	Define temporary VARIANT types as Globals, not as local variables (enabled by default).
Fill array in separate scopes	Fill in each array in a separate scope (enabled by default).
Fill structure in separate scopes	Fill in each structure in a separate scope (enabled by default).
Generate COM exceptions	Generate COM functions and methods that raised exceptions during recording (disabled by default).
Generate COM statistics	Generate recording time performance statistics and summary information (disabled by default).

Limit size of SafeArray log	Limit the number of elements printed in the safearray log per COM call, to 16 (enabled by default).
Release COM Objects	Record the releasing of COM objects when they are no longer in use (enabled by default).
Save Recordset content	Stores Recordset content as grids, to allow viewing of recordset in VuGen (enabled by default).
Trap binded moniker objects	Trap all of the bound moniker objects (disabled by default).

Correlations > Configuration Recording Options

This recording option pane enables you to configure settings for the Correlation tab.

To access	VuGen > Recording Options > Correlations > Configuration
Important information	<ul style="list-style-type: none"> • "Correlation Tab [Design Studio] Overview" on page 234 • "Correlations > Rules Recording Options" on page 151 • This recording option is only available for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.
Relevant tasks	"Correlate Scripts Using Design Studio" on page 250

User interface elements are described below:


UI Element	Description
Scan for correlations applying:	
Rules Scan	Apply correlation rules when performing correlation scan. For details, see "Correlations > Rules Recording Options" on page 151 .
Automatically correlate values found	Design Studio will automatically correlate dynamic values found using the rules scan.
Record Scan	Scan for correlations with the record based engine.
Replay Scan	Scan for correlations with the replay based engine.
Record and Replay scan configuration	

UI Element	Description
API used for correlations	<p>Select the API function to be used for correlation:</p> <p>Boundary based: web_reg_save_param_ex</p> <p>Regular Expression: web_reg_save_param_regexp</p> <div> <p>Note: If you change the API function, the changes will only take effect after a new scan.</p> </div>
Excluded strings	<p>Enables you to enter strings that should be ignored by the record and replay scan.</p> <p>For details, see "Exclude Strings or Content Types from the Correlation Scan" on page 241</p>
Excluded content types	<p>Enables you to enter content types that should be ignored by the record and replay scan. For details, see "Exclude Strings or Content Types from the Correlation Scan" on page 241.</p>
Ignore values shorter than []	<p>Enables you to define how short a dynamic value can be before it is ignored by the record or replay scan.</p> <p>Default length is 4 characters.</p>
Ignore values longer than []	<p>Enables you to define how long a dynamic value can be before it is ignored by the record or replay scan.</p> <p>Default length is 400 characters.</p>
Warn me if the dynamic string size is greater than 10 KB	<p>Issues a warning if you try to correlate a string whose size is 10 KB or larger.</p>
Ignore case when searching for correlation values	<p>Disable case sensitivity during correlation scan.</p>
Record scan configuration	

UI Element	Description
Heuristic level	<p>Enables you to set the filter level that controls the amount of correlation results that are returned. The higher the filter level, the shorter the scan will take to run.</p> <p>High. Design Studio performs a detailed scan returning a highly refined result set.</p> <p>Medium. Design Studio performs a less detailed scan returning more results. This is the default setting.</p> <p>Low. Design Studio performs a more detailed scan returning the most results. This setting may produce many unwanted results.</p>
Replay scan configuration	
Scan for differences between snapshots using	<p>Select a comparison method:</p> <ul style="list-style-type: none"> • HTML Comparison. Display the differences in HTML code only. • Text Comparison. Display all text, HTML, and binary differences.







Correlations > Rules Recording Options

This dialog box enables you to manage correlation rules that automatically correlate dynamic values during code generation. You can new and delete applications. You can also as add, delete, import, export and test correlation rules.

To access	<p>Do one of the following:</p> <ul style="list-style-type: none"> • Record > Recording Options > Correlations > Rules •  Design Studio > Options > Correlations tab > Rules
Important information	<ul style="list-style-type: none"> • "Correlation Overview" on page 232 • "Correlation Tab [Design Studio] Overview" on page 234
Relevant tasks	<p>"Correlate Scripts Using Design Studio" on page 250</p>

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Application List >	<p>A list of applications and their rules.</p> <ul style="list-style-type: none"> • Select the check box adjacent to the application to activate it during recording. • Expand the application tree to select the check box adjacent to the rules to activate them during recording.


UI Element	Description
	Add a new application to <Application List> .
	Enter a new rule for the selected application in Correlation Rules. For details, see "New Rule Pane" below .
	Delete the selected application or rule from the list.
	Import a file containing correlation rule definitions.
	Export a file containing a correlation rule definition.
	Test a correlation rule. For details, see "Token Substitution Testpad Dialog Box" on page 155 .

New Rule Pane

Enables you to define a new custom rule.

To access	Record > Recording Options > Correlation > Rules > New Rule
Important information	This pane is available only for specific protocols. For a complete list of protocols and their associated nodes, see the "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
	Opens the "Advanced Correlation Properties Dialog Box" on page 154 .

Action	<p>Specify the type of action for the rule from the following options:</p> <ul style="list-style-type: none">• Search for Parameters in all of the Body Text. Searches the entire body—not just links, form actions or cookies. It searches the text for a match using the borders that you specify.• Search for parameters inlinks and form actions. Searches within links and forms' actions for the text to parameterize. This method is for application servers where you know the context rules. You define a left boundary, a right boundary, an alternate right boundary, and an instance of the left boundary within the current link.• Search for Parameters from cookie headers. Similar to the previous rule, except that the value is extracted from cookie text (exactly as it appears in the recording log) instead of from a link or form action.• Parameterize form field value. Saves the named form field value to a parameter. It creates a parameter and places it in the script before the form's action step. For this option, you need to specify the field name.• Text to enter a web_reg_add_cookie function by method inserts a web_reg_add_cookie function if it detects a certain string in the buffer. It only adds the function for those cookies with the specified prefix. For this option, you need to specify the search text and the cookie prefix.
---------------	--

<p>Scan Type</p>	<p>The scan type. Select a scan type, and fill in the relevant details:</p> <p>Regular Expression:</p> <ul style="list-style-type: none"> • RegExp String. A regular expression to which this rule will apply. <p>Boundary Based:</p> <ul style="list-style-type: none"> • Left boundary. The left-most boundary where the rule will apply. • Right boundary. The right-most boundary where the rule will apply. Use the drop-down menu to define this boundary as either the end of a string, a newline character, or a user-defined text. • Match Case. Matches the case when looking for boundaries • Use '#' for any digit. Replaces all digits with a hash sign. The hash signs serve as wildcard, allowing you to find text strings with any digit. <p>Example: If you enable this option and specify HPE### as the left boundary, HPE193 and HPE284 will be valid matches.</p> <p>Attribute Based: Available for Search for Parameters in all of the Body Text actions only.</p> <ul style="list-style-type: none"> • TagName. The name of HTML element. • Extract. The name of HTML attribute whose value will be saved to the parameter. • Name/ID. The value of the HTML element's "name"/"Id" attributes. • Class. The value of the HTML element's "class" attribute. • Type. The value of the HTML element's "type" attribute. For example, for the INPUT element, Type can be "hidden". <p>XPath query:</p> <ul style="list-style-type: none"> • XPath string. An XPath expression to which this rule will apply. This element only applies to a XPath query scan type. When VuGen detects a value that matches this expression, it creates a web_reg_save_param_xpath function. <p>JSON query:</p> <ul style="list-style-type: none"> • JSONPath String. A JSON query expression to which this rule will apply. This element only applies to a JSON query scan type. When VuGen detects a value that matches this expression, it creates a web_reg_save_param_json function. For details about creating a JSON query, see Internet resources on JSONPath expressions.
<p>Parameter prefix</p>	<p>Uses a prefix in all automatically generated parameters based on this rule. Prefixes prevent you from overwriting existing user parameters. In addition, prefixes allow you to recognize the parameter in your script. For example, in Siebel Web, one of the built-in rules searches for the Siebel_row_id prefix.</p>

Advanced Correlation Properties Dialog Box

Enables you to set the advanced options for correlation rules.

To access	Record > Recording Options > Correlation > Rules > New Rule > Advanced
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see the "Protocol Compatibility Table" on page 204 .

User interface elements are described below:


UI Element	Description
Always create new parameter	Creates a new parameter for this rule even if the value replaced by the parameter has not changed from the previous instance.
Replace with parameter only for exact matches	Replaces a value with a parameter only when the text exactly matches the found value.
Reverse search	Performs a backwards search.
Left boundary instance	The number of occurrences of the left boundary in order for it to be considered a match.
Offset	The offset of the string within the found value.
Length	The length of the string, starting with the offset, to save to the parameter. If this is not specified, the parameter continues until the end of the found value.
Alternate right boundary	Alternative criteria for the right boundary, if the previously specified boundary is not found. Select one of the following options: User-defined Text, Newline Character, End Of Page .

Token Substitution Testpad Dialog Box

Enables you to test correlation rules before applying them.

To access	Record > Recording Options > Correlations > Rules > Test
------------------	---

User interface elements are described below:


UI Element	Description
	Runs the test.
Applied rules	A list of the rules and their values that were applied during the test.
Source string for substitution	Enter the source string for substitution.
Substitution Result	The results of the test.

Database > Database Recording Options

Enables you to set the recording options for database protocols.

To access	Record > Recording Options > Database > Database
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
	Opens the "Database > Advanced Recording Options Dialog Box" below.
Automatic transactions	If enabled, every lrd_exec and lrd_fetch function is marked as a transaction with lr_start_transaction and lr_end_transaction functions. Default value: Disabled.
Script options	Generates comments into recorded scripts, describing the lrd_stmt option values. In addition, you can specify the maximum length of a line in the script. Default value: 80 characters.
Think time	VuGen automatically records the operator's think time. You can set a threshold level, below which the recorded think time will be ignored. If the recorded think time exceeds the threshold level, VuGen places an lr_think_time statement before LRD functions. If the recorded think time is below the threshold level, an lr_think_time statement is not generated. Default value: five seconds.

Database > Advanced Recording Options Dialog Box

Enables you set the advanced recording options for database protocols.

To access	Record > Recording Options > Database > Database > Advanced
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
------------	-------------


Recording log options	<p>You can set the detail level for the trace and ASCII log files.</p> <p>Trace file detail level:</p> <ul style="list-style-type: none"> • Off • Error Trace. Only logs error messages. • Brief Trace. Logs errors and lists the functions generated during recording. • Full Trace. Logs all messages, notifications, and warnings. <p>ASCII file detail level: Instructs VuGen to generate ASCII type logs of the recording session.</p> <ul style="list-style-type: none"> • Off • Brief detail. Logs all of the functions. • Full detail. Logs all of the generated functions and messages in ASCII code.
Code generation buffer size	<p>Specify in kilobytes the maximum size of the code generation buffer.</p> <p>Default value: 128 kilobytes.</p>








Data Format Extension > Chain Configuration Recording Options

Enables you to add, delete, and modify chains, and to manage the DFEs that are included in the chains.

To access	Record > Recording Options > Data Format Extension > Chain Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.
See also	<ul style="list-style-type: none"> • "Data Format Extensions (DFEs) - Overview" on page 704 • "Data Format Extension List" on page 713 • "Implementing GWT-DFE Support" on page 722 • "Add Prefix/Postfix to Chain Dialog Box" on the next page

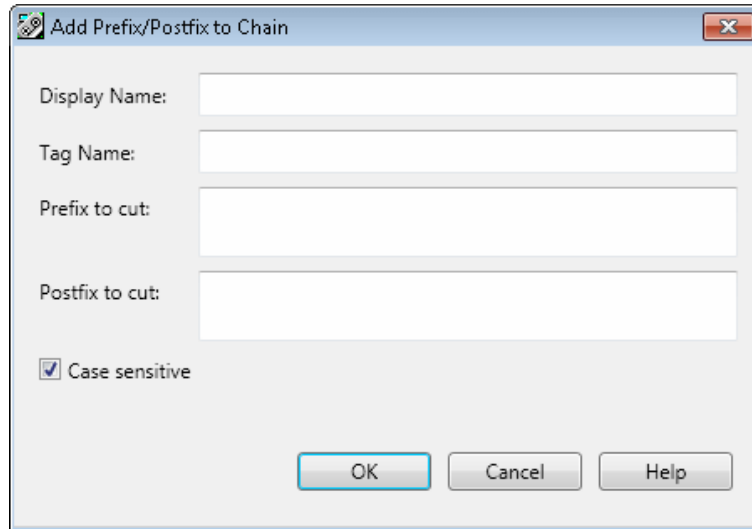
User interface elements are described below:


UI Element	Description
Chains pane	Displays a list of the DFE chains that are defined for the script.
	Add Chain. Enables you to add a new chain.

	Edit Chain Name. Enables you to modify the name of the selected chain.
	Delete Chain. Deletes the selected chain.
Chain: <chain name> pane	
	Add DFE. Enables you to add a DFE to the selected chain in the Chains pane. For more information on Data Format Extensions, see "Data Format Extension List" on page 713 .
	Edit DFE. For a <i>Prefix Postfix Extension</i> , edit the prefix and postfix to cut. For a <i>GWT Extension</i> , specify the classpath.
	Add Custom Path. Lets you add a custom classpath for GWT. You can use this to specify Linux paths. For example, /tmp/stockwatcher.war.
	Delete DFE. Deletes the selected DFE from the chain.
	Move Up/Down. Moves the selected Data Format Extension up or down in the chain. Extensions are run in the order in which they appear in the extensions list.
Name	The display name of the Data Format Extension.
Tag	The unique ID of the extension.
Provider	The creator of the Data Format Extension.
Continue Processing	<p>Determines how the chain behaves after the DFE is applied:</p> <ul style="list-style-type: none"> • True: The data is passed on to the next DFE in the chain, whether or not the data was converted. • False: If the DFE converted the data that it received, the chain is terminated - no further DFEs are applied to the data. If the DFE did not convert the data that it received, the data is passed on to the next DFE in the chain. <div style="border-left: 2px solid green; padding-left: 10px; margin-top: 10px;"> <p>Note: If the chain contains only a single DFE, the Continue Processing setting is not significant.</p> </div>

Add Prefix/Postfix to Chain Dialog Box

This dialog enables you to add or edit a prefix/postfix extension to the selected chain.



To access	<ol style="list-style-type: none"> 1. Go to Record > Recording Options > Data Format Extension > Chain Configuration node. 2. In the Chain: <Chain name> area, click the  button. 3. Select Prefix Postfix Extension and click OK.
------------------	---

User interface elements are described below:


UI Element	Description
Case sensitive	Sets the extension to cut from the defined prefix and postfix of the string only if the letter cases match.
Display name	The name of Prefix/Postfix Extension.
Postfix to cut	The section you want to cut, from the end of the string.
Prefix to cut	The section you want to cut, from the beginning of the string.
Tag name	The unique ID of the Prefix/Postfix Extension.

See also:

- [Data Format Extension > Chain Configuration Node](#)
- [Data Format Extensions](#)
- [Data Format Extension List](#)

Add Data Format Extension

This dialog box enables you to select the data format extension type.


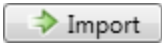
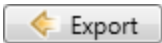
To access	VuGen > Recording Options > Data Format Extension > Chain Configuration > 
Important information	<ul style="list-style-type: none"> • "Data Format Extension > Chain Configuration Recording Options" on page 157 • "Add Prefix/Postfix to Chain Dialog Box" on page 158
Relevant tasks	"Apply DFE Chains to Sections of the HTTP Message" on page 711
Data Format Extension	Description
Base64 Extension	Decodes strings that are encoded with a Base64 encoder.
GWT Extension	Transforms GWT data to XML format.
URL Encoding Extension	Decodes strings that are encoded with URL encoding format.
JSON to XML Extension	Transforms JSON data to XML format.
XML Extension	Receives data and checks to see if it conforms with XML syntax. This check allows VuGen to perform correlations based on XPath and to display snapshot data in an XML viewer.
Prefix Postfix Extension	Enables you to cut data from the beginning and/or end of a string which you do not want decoded. You can add and customize as many prefix/postfix extensions as required. Each postfix/prefix extension created should have a unique display name and tag name.
Binary to XML Extension	Transforms Microsoft WCF binary XML into XML format.
Remedy to XML Extension	<p>Transforms Remedy request data into XML format.</p> <p>This extension does not transform Remedy response data, which is JavaScript code.</p>
XSS Extension	Enables you to test sites that use Cross Site Scripting (XSS) defense code.




Data Format Extension > Code Generation Recording Options

Enables Data Format Extensions during code generation, and enables you to define chains for each section of the HTTP message.

To access	Record > Recording Options > Data Format Extension > Code Generation
Important information	This node is available for specific protocols only. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .
See also	<ul style="list-style-type: none"> • "Data Format Extensions (DFEs) - Overview" on page 704 • "Data Format Extension List" on page 713

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
Enable data format extension	Enables you to select chains for each message section of the HTTP message. By default, this option is not selected.
Configuration	
Format	<ul style="list-style-type: none"> • Code and snapshots. (default) Enables Data Format Extensions on the code and snapshot data. • Snapshots. Enables Data Format Extensions on snapshot data, but does not format the data in the script itself.
Verify formatted data	<p>Checks the results of the formatted data by converting it back to the original state and verifying that it matches the original data.</p> <div>  Note: Available for Base64 extension only. </div>
Chain Assignment	
	Imports the Data Format Extensions from a file.
	Exports the Data Format Extensions to a file.

<Message sections list>	<p>Displays a list of the following sections of the HTTP message included in the script:</p> <ul style="list-style-type: none"> • Body • Headers • Cookies • Query String <p>When you select a message section from the list, the title of the section chains pane (described below) reflects your selection and the pane displays the list of chains for that section.</p>
<Section Chains>	
	<p>Add Chain. Adds chain to selected message section.</p> <div data-bbox="422 716 1412 1016"> <p>Note:</p> <ul style="list-style-type: none"> • Enabled for Headers and Cookies sections only. Enables you to add additional chains to the selected message section. • For VuGen to correctly match the chain to the Headers or Cookies section, the name in the Name column must match the name of the Headers or Cookies section. </div>
	<p>Delete Chain. Removes chain from corresponding message section.</p> <div data-bbox="422 1106 1412 1234"> <p>Note: You cannot delete the default options from any of the message sections.</p> </div>
	<p>Reset. Clears the selected chain in the Chain column.</p>

Flex > RTMP Recording Options

This node enables you to include the flex_rtmp_recive_stream step in Flex RTMP scripts.

To access	Record > Recording Options > Flex > RTMP
Important Information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.

User interface elements are described below:

UI Element	Description
------------	-------------

Generate single step for RTMP/T stream handling	Generates a single step, flex_rtmp_receive_stream , when recording a stream. This step does not replay certain actions, such as pause and seek. If your script requires these actions, clear the check box to record all receive and send steps.
--	---

Flex > Configuration Recording Options

Enables you to set an external JVM (Java Virtual Machine) path.

To access	Record > Recording Options> Flex > Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Use External JVM	<p>Enables you to use an external JVM. If you choose this option, you must specify the path:</p> <p>External JVM Path The full path of the external JVM.</p> <p>VuGen must be restarted for the changes to be applied.</p>
UseGraniteDS configuration	<p>Defines the server side Data Service configuration.</p> <p>If you select this option, do not select Use Flex LCDS/BlazeDS jars to serialize the messages. Ensure that the granit-config.xml file matches the one deployed on the server.</p>
Maximum Formatted Request/Response size to print	<p>Enables you to specify the maximum character length of a request or response body to be captured in the log files. The option only affects the flex_amf_call and flex_remoting_call steps.</p> <p>For example, if you specify a value of 1048576 characters (1MB), only responses or requests with a length less than a megabyte will be printed on the Code Generation or Replay log.</p>





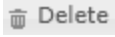
Flex > Externalizable Objects Recording Options

This dialog box enables you to configure how to handle externalizable objects in Flex scripts.

To access	Record > Recording Options > Flex > Externalizable Objects
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

Relevant tasks	<ul style="list-style-type: none"> • How to Serialize Using External Java Serializer • "Serialize Flex Scripts" on page 512
-----------------------	---

User interface elements are described below:

UI Element	Description
Do not serialize externalizable objects	Generate script using default settings.
Serialize objects using	<p>Select the appropriate option:</p> <ul style="list-style-type: none"> • Select LoadRunner AMF serializer if you are not using the Adobe LiveCycle Data Services or Adobe BlazeDS server. • Select Custom Java classes and select one or both of the available options: <ul style="list-style-type: none"> • Select Use Flex LCDS/BlazeDS jars if you are using Flex LCDS or BlazeDS jars to serialize the messages. If you selected UseGraniteDS configuration in the Configuration node, do not select Use Flex LCDS/BlazeDS jars. • Select Use additional jars to add additional jars to serialize the messages. You must copy the jar files from the server and specify their location in the Classpath Entries list described below. Copy only those jars that contain the class that is externalizable. Ensure that the files exist in the same location on all load generator computers. If you add jars with the same names as the Flex LCDS or Blaze DS jars chosen by selecting the first check box, these files will be overwritten.
Classpath Entries List	
	Down Arrow. Moves a classpath entry down the list.
	Up Arrow. Moves a classpath entry up the list.
	Add Classpath. Adds a new line to the classpath list.
	Add Classpath Folder. Adds all files from the folder to the classpath list.
	Delete. Permanently removes a classpath.

See also:

- [Flex Overview](#)
- [Externalizable Objects in Flex Scripts](#)

FTP > Configuration Recording Options

Enables you to indicate additional parameters to use when recording FTP applications.

To access	Record > Recording Options > FTP > Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Welcome messages	<p>In order for VuGen to record FTP traffic, it must detect an FTP welcome message. The default welcome messages contain these strings: FTP, ncFTP, and proFTPDA.</p> <p>This setting lets you specify a custom list of welcome messages issued by an FTP server when you connect. The recording of the FTP traffic only begins after the welcome message is detected.</p> <p>You can specify entire words or word fragments and all types of characters. Use the pipe " " symbol as a separator between the strings.</p>


General > Code Generation Recording Options

This pane of the Recording Options dialog box enables you to define what tasks VuGen performs automatically after generating a Vuser script.

To access	Record > Recording Options > General > Code Generation
Relevant tasks	<p>"Create an Asynchronous Vuser Script" on page 381</p> <p>"Correlate Scripts Using Design Studio" on page 250</p>

User interface elements are described below:

UI Element	Description
Correlations Scan	Instructs VuGen to analyze the Vuser script to locate dynamic values that may need to be correlated. This scan is performed after a new script is recorded and after an existing script is regenerated.
Async Scan	Instructs VuGen to analyze the Vuser script to locate asynchronous communication. This scan is performed after a new script is generated and after an existing script is regenerated.

UI Element	Description
Async Options...	Opens the "Asynchronous Options Dialog Box" on page 406.
	 Tip: We recommend that you use a 100% display size.

General > Protocol Recording Options

Enables you to set the script generation preferences by setting the scripting language and options.

To access	Record > Recording Options > General > Protocols
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.

User interface elements are described below:



UI Element	Description
Active Protocols List	A list of the protocols which comprise your multiple protocol script. VuGen lets you modify the protocol list for which to generate code during the recording session. Select the check boxes adjacent to the protocols you want to record in the next recording session. Clear the check boxes adjacent to the protocols you do not want to record in the next recording session.

General > Recording - Recording Options

Enables you to specify what information to record and which functions to use when generating a Vuser script, by selecting a recording level.

To access	Record > Recording Options > General > Recording
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.

User interface elements are described below:

UI Element	Description
	Opens the "Advanced HTML Dialog Box" on page 168.
	Opens the "Advanced URL Dialog Box" on the next page.


HTML-based script	This is the default recording level for Web - HTTP/HTML Vusers. It instructs VuGen to record HTML actions in the context of the current Web page. It does not record all resources during the recording session, but downloads them during replay.
URL-based script	Record all requests and resources from the server. It automatically records every HTTP resource as URL steps (web_url statements), or in the case of forms, as web_submit_data . It does not generate the web_link , web_image , and web_submit_form functions, nor does it record frames. We recommend this option for non-browser applications.

Advanced URL Dialog Box

Enables you to set the advanced options for scripts using the URL recording mode.

To access	Record > Recording Options > General > Recording > URL Advanced
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
	Restores the default settings of this dialog box.
Create concurrent groups for resources after their source HTML page	Records the resources in a concurrent group (enclosed by web_concurrent_start and web_concurrent_end statements) after the URL. Resources include files such as images and js files. If you disable this option, the resources are listed as separate web_url steps, but not marked as a concurrent group.
Enable EUC-Encoded Web Pages	(For Japanese windows only) Instructs VuGen to use EUC encoding. For more information, see "EUC-Encoding (Japanese Windows only)" on page 209 .
Use web_custom_request only	Records all HTTP requests as custom requests. VuGen generates a web_custom_request function for all requests, regardless of their content. Recommended for non-browser applications.

Note: The functions included within the concurrent group are not executed immediately. Instead, they are registered for concurrent execution. When the concurrent group is closed, all of the functions registered as concurrent are executed in parallel. The maximum number of


items submitted in parallel depends on the browser being emulated. For example with Internet Explorer 9, up to six resources are downloaded simultaneously.

Advanced HTML Dialog Box

Enables you to set the advanced options for HTTP-based scripts.

To access	Record > Recording Options > General > Recording > HTML Advanced
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
	Restores the default settings of this dialog box.
Non-HTML generated elements	<p>Many Web pages contain non-HTML elements, such as applets, XML, ActiveX elements, or JavaScript. These non-HTML elements usually contain or retrieve their own resources. Using the following options, you can control how VuGen records non HTML-generated elements.</p> <ul style="list-style-type: none"> • Record within the current script step. Does not generate a new function for each of the non HTML-generated resources. It lists all resources as arguments of the relevant functions, such as web_url, web_link, and web_submit_data. The resources, arguments of the Web functions, are indicated by the EXTRARES flag. • Record in separate steps and use concurrent groups. Creates a new function for each one of the non HTML-generated resources and does not include them as items in the page's functions (such as web_url and web_link). All of the web_url functions generated for a resource are placed in a concurrent group (surrounded by web_concurrent_start and web_concurrent_end). • Do not record. Does not record any non-HTML generated resources.
Script type	<ul style="list-style-type: none"> • A script describing user actions. Generates functions that correspond directly to the action taken. It creates URL (web_url), link (web_link), image (web_image), and form submission (web_submit_form) functions. The resulting script is very intuitive and resembles a context sensitive recording. • A script containing explicit URL's only. Records all links, images and URLs as web_url statements, or in the case of forms, as web_submit_data. It does not generate the web_link, web_image, and web_submit_form functions. The resulting script is less intuitive. This mode is useful for instances where many links within your site have the same link text. If you record the site using the first option, it records an ordinal (instance) for the link, but if you record using the second option, each link is listed by its URL. This facilitates parameterization and correlation for that step.

General > Script Recording Options

Enables you to set the script generation preferences by setting the scripting language and options.

To access	Record > Recording Options > General > Script
Important information	<p>This node is available for specific protocols only. In addition, the list of options differs between protocols.</p> <p>For a complete list of protocols and their associated nodes, see the "Protocol Compatibility Table" on page 204.</p>

User interface elements are described below:

UI Element	Description
Scripting Language	<p>Select the language in which to generate the Vuser script, C or JavaScript. The default is C.</p> <div> <p>Note: This option is available for Web - HTTP/HTML Vuser scripts only.</p> </div>
Add a comment for each action	<p>Insert informative logging messages before each message invocation (non-C only). Default value: enabled.</p>
Close all AUT processes when recording stops	<p>Automatically closes all of the AUT's (Application Under Test) processes when VuGen stops recording. Default value: disabled.</p>
Correlate arrays	<p>Tracks and correlates arrays of all data types, such as string, structures, numbers, and so on. Default value: enabled.</p>
Correlate large numbers	<p>Correlates long data types such as integers, long integers, 64-bit characters, float, and double. Default value: disabled.</p>
Correlate simple strings	<p>Correlates simple, non-array strings and phrases. Default value: disabled.</p>
Correlate small numbers	<p>Correlates short data types such as bytes, characters, and short integers. Default value: disabled.</p>

UI Element	Description
Correlate structures	Tracks and correlates complex structures. Default value: enabled.
Declare primitives as locals	Declares primitive value variables as local variables rather than class variables (C, C#, and .NET only). Default value: enabled.
Explicit variant declaration	Declares variant types explicitly in order to handle ByRef variants (Visual Basic for Applications only). Default value: enabled.
Generate fixed think time after end transaction	Adds a fixed think time, in seconds, after the end of each transaction. When you enable this option, you can specify a value for the think time. Default value: disabled, 3 seconds when enabled.
Generate recorded events log	Generates a log of all events that took place during recording. Default value: disabled.
Generate think time greater than threshold	Uses a threshold value for think time. If the recorded think time is less than the threshold, VuGen does not generate a think time statement. You also specify the threshold value. The default values is 3—if the think time is less than 3 seconds, VuGen does not generate think time statements. If you disable this option, VuGen will not generate any think times. Default value: enabled, 3 seconds.
Insert output parameters values	Inserts output parameter values after each call (C, C#, and .NET only). Default value: disabled.
Insert post-invocation info	Insert informative logging messages after each message invocation (non-C only). Default value: enabled.
Maximum number of lines in action file	Create a new file if the number of lines in the action exceeds the specified threshold. The default threshold is 60000 lines (C, C#, and .NET only). Default value: disabled.

UI Element	Description
Replace long strings with parameter	<p>Save strings exceeding the maximum length to a parameter. This option has an initial maximum length of 100 characters. The parameters and the complete strings are stored in the lr_strings.h file in the script's folder in the following format: const char <paramName_uniqueID> ="string".</p> <p>This option allows you to have a more readable script. It does not effect the performance of the script. Default value: enabled.</p>
Reuse variables for primitive return values	<p>Reuse the same variables for primitives received from method calls. This overrides the Declare primitives as locals setting. Default value: enabled.</p>
Track processes created as COM local servers	<p>Track the activity of the recorded application if one of its sub-processes was created as a COM local server (C and COM only). Default value: enabled.</p>
Use full type names	<p>Use the full type name when declaring a new variable (C# and .NET only). Default value: disabled.</p>
Use helpers for arrays	<p>Use helper functions to extract components in variant arrays (Java and VB Scripting only). Default value: disabled.</p>
Use helpers for objects	<p>Use helper functions to extract object references from variants when passed as function arguments (Java and VB Scripting only). Default value: disabled.</p>
Use protected application recording	<p>Use this option if VuGen is unable to record your application. Your application may block access to VuGen, and recording with this option selected may enable access. Default value: disabled.</p>
Warn me if the application being recorded encounters an error	<p>Selecting this option enables VuGen to prompt you to cancel the recording if the recorded application crashes or if no events are recorded for 3 minutes. If you choose to cancel the recording, no script is generated. In proxy mode, if the recorded application crashes, there is no error message. Instead, a no-events error is displayed after 3 minutes.</p> <p>Default value: enabled.</p> <div> <p>Note: This option is available for Web - HTTP/HTML Vuser scripts only.</p> </div>

GUI Properties > Web Event Configuration Recording Options

Enables you to set the level of detail recorded in a script (web event recording).

To access	Record > Recording Options > GUI Properties > Web Event Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Basic Event Configuration Level	<ul style="list-style-type: none"> Always records click events on standard Web objects such as images, buttons, and radio buttons. Always records the submit event within forms. Records click events on other objects with a handler or behavior connected. Records the mouseover event on images and image maps only if the event following the mouseover is performed on the same object.
Custom Settings	Opens the "Custom Web Event Recording Configuration Dialog Box" below, where you can customize the event recording configuration.
High Event Configuration Level	In addition to the objects recorded in the Medium level, it records mouseover, mousedown, and double-click events on objects with handlers or behaviors attached.
Medium Event Configuration Level	In addition to the objects recorded in the Basic level, it records click events on the <DIV>, , and <TD> HTML tag objects.

Custom Web Event Recording Configuration Dialog Box

Enables you to customize the level of web event recording.

To access	Record > Recording Options > GUI Properties > Web Event Configuration > Custom Settings
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Object List>	A list of the web objects. Each web object can be customized according to the other settings in this dialog box.
<Object Menu>	<ul style="list-style-type: none"> • Add. Adds a new HTML tag object to the object list. Type in the name of the tag. • Delete. Deletes an object from the object list.
Event Menu	<ul style="list-style-type: none"> • Add. Adds an event to the Event Name column of this object. • Delete. Deletes an event from the Event Name column of this object.
Event Name	A list of events associated with the object.
File Menu	<ul style="list-style-type: none"> • Load Configuration. Loads a previously created custom configuration. • Save Configuration As. Saves the current configuration.
Listen	<p>The criteria which determines when VuGen listens for an event.</p> <ul style="list-style-type: none"> • Always. Always listen to the event. • If Handler. Listens to the event if a handler is attached to it. A handler is code in a Web page, typically a function or routine written in a scripting language, that receives control when the corresponding event occurs. • If Behavior. Listens to the event if a DHTML behavior is attached to it. A DHTML behavior encapsulates specific functionality or behavior on a page. When applied to a standard HTML element on a page, a behavior enhances that element's default behavior. • If Handler or Behavior. Listens to the event if either a handler or a behavior is attached to it. • Never. Never listens to the event. <p>For more information, see "Tips for Working with Event Listening and Recording" on page 212.</p>

UI Element	Description
Record	<p>The criteria which determines when VuGen records an event.</p> <ul style="list-style-type: none"> • Enabled. Records the event each time it occurs on the object as long as VuGen listens to the event on the selected object, or on another object to which the event bubbles. Bubbling is the process whereby, when an event occurs on a child object, the event can travel up the chain of hierarchy within the HTML code until it encounters an event handler to process the event. • Disabled. Does not record the specified event and ignores event bubbling where applicable. • Enabled on next event. Same as Enabled, except that it records the event only if the subsequent event occurs on the same object. For example, suppose a mouseover behavior modifies an image link. You may not want to record the mouseover event each time you happen to move the mouse over this image. Because only the image that is displayed after the mouseover event enables the link event, however, it is essential that the mouseover event is recorded before a click event on the same object. <p>For more information, see "Tips for Working with Event Listening and Recording" on page 212.</p>
Reset Settings	Resets the custom settings to the settings of your choice: basic , medium , or high .

GUI Properties > Advanced Recording Options

Enables you to set advanced recording options for Click & Script Vusers.

To access	Record > Recording Options > GUI Properties > Advanced
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.

User interface elements are described below:

Recording Settings Properties

UI Element	Description
Record rendering-related property values	<p>Records the values of the rendering-related properties of DOM objects (for example, offsetTop), so that they can be used during replay. Note that this may significantly decrease the replay speed.</p> <p>Default value: disabled.</p>

Record 'click' by mouse events	Records mouse clicks by capturing mouse events instead of capturing the clickO method. Enable when the recorded application uses the DOM clickO method, to prevent the generation of multiple functions for the same user action. Default value: enabled.
Record socket level data	Enables the recording of socket level data. If you disable this option you will need to manually add the starting URL before recording. In addition, you will be unable to regenerate the script on an HTML level. Default value: enabled.
Generate snapshots for Ajax steps	Enables generation of snapshots for Ajax steps. Enabling this option can result in errors during recording. Default value: disabled.

Code Generation Settings Properties






UI Element	Description
Enable generation of out-of-context steps	Creates a URL-based script for ActiveX controls and Java applets, so that they will be replayed. Since these functions are not part of the native recording, they are referred to as out-of-context recording. Default value: disabled.
Enable automatic browser title verification	Enables automatic browser title verification. Default value: disabled.
Perform a title verification for	<ul style="list-style-type: none"> • each navigation. Performs a title verification only after a navigation. When a user performs several operations on the same page, such as filling out a multi-field form, the title remains the same and verification is not required. • each step. Performs a title verification for each step to make sure that no step modified the browser title. A modified browser title may cause the script to fail. • Perform a title verification using the URL if the title is missing. For browser windows without a title, perform a title verification for each step using its URL.

HTTP Properties > Advanced Recording Options

Enables you to customize the code generation settings in the area of think time, resetting contexts, saving snapshots, generating **web_reg_find** functions, and masking passwords.

To access	Record > Recording Options > HTTP Properties > Advanced
Important information	<ul style="list-style-type: none"> • This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204. • Some options in this node are not available in multi-protocol scripts.

User interface elements are described below:

UI Element	Description
	Opens the "Headers Dialog Box" on page 179.
	Opens the "Content Type Filters Dialog Box" on page 180.
	Opens the "Non-Resources Dialog Box" on page 181.
Reset context for each action	<p>Resets all HTTP contexts between actions. Resetting contexts allows the Vuser to more accurately emulate a new user beginning a browsing session. This option resets the HTML context, so that a context-less function is always recorded in the beginning of the action. It also clears the cache and resets the user names and passwords.</p> <p> Note: Available only for Web and Oracle NCA protocols</p>
Save snapshot resources locally	Saves a local copy of the snapshot resources during record and replay, thereby creating snapshots more accurately and displaying them quicker.
Generate web_reg_find functions for page titles	<p>Generates web_reg_find functions for all HTML page titles. VuGen adds the string from the page's title tag and uses it as an argument for web_reg_find.</p> <ul style="list-style-type: none"> • Generate web_reg_find functions for sub-frames. Generates web_reg_find functions for page titles in all sub-frames of the recorded page. <p> Note: Available only for Web and Oracle NCA protocols</p>
Add comment to script for HTTP errors while recording	Adds a comment to the script for each HTTP request error. An error request is defined as one that generated a server response value of 400 or greater during recording.

Support charset	<ul style="list-style-type: none">• UTF-8. Enables support for UTF-8 encoding. This instructs VuGen to convert non-ASCII UTF-8 characters to the encoding of your locale's machine in order to display them properly in VuGen. You should enable this option only on non-English UTF-8 encoded pages. The recorded site's language must match the operating system language. You cannot record non-English Web pages with different encodings (for example, UTF-8 together with ISO-8859-1 or shift_jis) within the same script.• EUC-JP. If you are using Japanese Windows, select this option to enable support for Web sites that use EUC-JP character encoding. This instructs VuGen to convert EUC-JP strings to the encoding of your locale's machine in order to display them properly in VuGen. VuGen converts all EUC-JP (Japanese UNIX) strings to the SJIS (Japanese Windows) encoding of your locale's machine, and adds a web_sjis_to_euc_param function to the script. (Kanji only)
------------------------	---

Parameterize server names	<p>(Web - HTTP/HTML only) VuGen identifies server names and IP addresses when you regenerate a Vuser script. These server names and IP addresses are contained in specific arguments associated with specific functions in the Vuser script. [See the table below for details.] When this option is enabled, VuGen replaces the identified server names and IP addresses with parameters. Parameterizing server names and IP addresses enables you to run the Vuser script in different environments by simply changing the server and IP address values in the parameter file. For an introduction to parameters, see "Parameterization Overview" on page 342.</p> <div style="background-color: #e6f2e6; padding: 10px; border: 1px solid #ccc;"> <p>Note: To identify data for parameterization, VuGen searches the arguments that are listed for the following functions:</p> </div> <table border="1" data-bbox="477 716 1409 1440"> <thead> <tr> <th>API Function</th><th>Arguments</th></tr> </thead> <tbody> <tr> <td>web_url</td><td> <ul style="list-style-type: none"> URL Referrer </td></tr> <tr> <td>web_custom_request</td><td> <ul style="list-style-type: none"> URL Referrer </td></tr> <tr> <td>web_image</td><td> <ul style="list-style-type: none"> URL Referrer </td></tr> <tr> <td>web_submit_data</td><td> <ul style="list-style-type: none"> Action URL Referrer </td></tr> <tr> <td>web_submit_form</td><td> <ul style="list-style-type: none"> Action URL Referrer </td></tr> </tbody> </table> <p>By default, this option is not selected.</p>	API Function	Arguments	web_url	<ul style="list-style-type: none"> URL Referrer 	web_custom_request	<ul style="list-style-type: none"> URL Referrer 	web_image	<ul style="list-style-type: none"> URL Referrer 	web_submit_data	<ul style="list-style-type: none"> Action URL Referrer 	web_submit_form	<ul style="list-style-type: none"> Action URL Referrer
API Function	Arguments												
web_url	<ul style="list-style-type: none"> URL Referrer 												
web_custom_request	<ul style="list-style-type: none"> URL Referrer 												
web_image	<ul style="list-style-type: none"> URL Referrer 												
web_submit_data	<ul style="list-style-type: none"> Action URL Referrer 												
web_submit_form	<ul style="list-style-type: none"> Action URL Referrer 												
Generate steps with missing responses	<p>Generate steps for HTTP requests that are missing server responses.</p>												
Generate web_add_cookie functions	<p>Detect the time when a cookie is created, and generate web_add_cookie or web_add_cookie_ex functions. If you clear this option, the above functions will not appear in the script.</p>												
Generate steps for WebSocket traffic	<p>Generate code and correlate WebSocket functions. If you clear this option, WebSocket functions will not appear in the script.</p>												

Replace passwords with masked parameters	When generating a script, replace actual passwords with a masked string.
Generate API calls for specific HTTP status codes	Generates API calls for the specified custom (non-standard) HTTP status codes. You should separate multiple entries with semicolons. For example: 302;303;304
Use the LoadRunner Proxy to record a local application	Provides an alternative way to record if the standard VuGen recording mechanism is not compatible with your application. This applies when you have selected to record a Web browser or Windows application. For details, see "Record a Script via a Proxy" on page 216 . Note: After recording, clear this option to restore the default mode.
Use streaming mode when recording with the LoadRunner Proxy	Streaming mode enables HTTP data portions received from the server to be forwarded to the application with buffering. This allows you to record asynchronous push communication. Note: If this option is enabled, the remote recording toolbar is disabled.



Headers Dialog Box

Enables you to automatically send additional HTTP headers with every HTTP request submitted to the server.

To access	Record > Recording Options > HTTP > Advanced > Headers
Important information	<ul style="list-style-type: none"> This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204. The following standard headers are considered risky: Authorization, Connection, Content-Length, Cookie, Host, If-Modified-Since, Proxy-Authenticate, Proxy-Authorization, Proxy-Connection, Referer, and WWW-Authenticate. They are not recorded unless selected in the Header list.

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Plus. Adds a new entry.
	Minus. Deletes an entry.



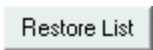
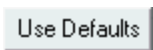
	Restores the current list to the default values and entries.
	Restores all lists to the default values and entries.
<Drop-down menu>	Controls the options for this dialog box: <ul style="list-style-type: none"> • Do not record headers • Record headers in list • Record headers not in list
<Header list>	List of headers which may or may not be recorded. The lists vary depending on which drop-down item is selected. Each item can be selected or deselected using its individual check box.

Content Type Filters Dialog Box

Enables you to filter content types for your recorded script. You can specify the type of the content you want to record or exclude from your script.

To access	Record > Recording Options > HTTP > Advanced > Content Types
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
	Plus. Adds a new entry.
	Minus. Deletes an entry.
	Restores the current list to the default values and entries.
	Restores all lists to the default values and entries.
<Drop-down menu>	Controls the options for this dialog box: <ul style="list-style-type: none"> • Do not filter content types. • Filter content types in list. • Filter content types not in list.
<Header list>	List of content types which may or may not be filtered. The lists vary depending on which drop-down item is selected. Each item can be selected or deselected using its individual check box.



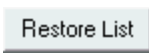
Non-Resources Dialog Box

When you record a script, VuGen indicates whether or not it will retrieve the resource during replay using the Resource attribute in the **web_url** function. If the Resource attribute is set to 0, the resource is retrieved during script execution. If the Resource attribute is set to 1, the Vuser skips the resource type.

You can exclude specific content types from being handled as resources. For example, you can indicate to VuGen that **gif** type resources should not be handled as a resource and therefore be downloaded unconditionally.

To access	Record > Recording Options > HTTP > Advanced > Non-Resources
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Add. Adds a new entry to the list.
	Remove. Deletes an entry from the list.
	Restores the default list.
<Non-Resource Content Type list>	List of items which should not be recorded as resources. Each item can be selected or deselected using its individual check box.

Java > VM Recording Options

Enables you to indicate additional parameters to use when recording Java applications.

To access	Record > Recording Options > Java Environment Settings > Java VM
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Additional VM Parameters	List the Java command line parameters here. These parameters may be any Java VM argument. The common arguments are the debug flag (-verbose) or memory settings (-ms , -mx). In addition, you may also pass properties to Java applications in the form of a -D flag. For more information about the Java VM flags, see the JVM documentation.




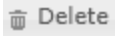
Prepend CLASSPATH to - Xbootclasspath parameter	Instructs VuGen to add the Classpath before the Xbootclasspath (prepend the string).
Use classic Java VM	Instructs VuGen to use the classic version of VM (for example, not Sun's Java HotSpot).
Use the specified Additional VM Parameters during replay	Instructs VuGen to use the same Additional VM parameters in replay.

Java > Classpath Recording Options

Enables you to specify the location of additional classes that were not included in the system's classpath environment variable. You may need these classes to run Java applications and insure proper recording.

To access	Record > Recording Options > Java Environment Settings > Classpath
Important information	For Java 8 and later. For the " Java Record Replay Protocol " on page 516, the classpaths are recorded when the script is recorded. Check that no unnecessary paths are recorded and that all necessary paths are recorded correctly.

User interface elements are described below:

UI Element	Description
	Down Arrow. Moves a classpath entry down the list.
	Up Arrow. Moves a classpath entry up the list.
	Add Classpath. Adds a new line to the classpath list.
	Delete. Permanently removes a classpath.
Classpath Entries List	A list of classpath entries.


Microsoft .NET > Recording - Recording Options

This screen enables you to set the recording options for .NET Vuser scripts.

To access	Record > Recording Options > Microsoft .NET > Recording
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Code Generation	<p>Allow you to indicate whether to show warnings, a stack trace, or all event subscriptions during code generation.</p> <ul style="list-style-type: none"> • Show Warnings. Shows warning messages that are issued during the code generation process. • Show Stack Trace. Shows the recorded stack trace if it is available. • Show All Event Subscriptions. Generates code for all event subscriptions that were recorded. If this option is disabled, VuGen will only generate code for events in which both the publisher (the object which invokes the event) and the subscriber (the object informed of the event) are included in the filter. <p>Default value: disabled.</p>
Debug Options	<p>Enables you to trace the stack and specify its size.</p> <ul style="list-style-type: none"> • Stack Trace. Traces the contents of the stack for each invocation within the script. It allows you to determine which classes and methods were used by your application. This can be useful in determining which references, namespaces, classes, or methods to include in your filter. Enabling the trace may affect your application's performance during recording. <p>Default value: disabled</p> <ul style="list-style-type: none"> • Stack Trace Limit. The maximum number of calls to be stored in the stack. If the number of calls exceeds the limit, VuGen truncates it. <p>Default value: 20 calls.</p>
Filters	<ul style="list-style-type: none"> • Ignore all assemblies by default. Ignores all assemblies that are not explicitly included by the selected filter. If you disable this option, VuGen looks for a matching filter rule for all assemblies loaded during the recording.

Logging	<p>The Logging options let you set the level of detail that is recorded in the recording log file.</p> <ul style="list-style-type: none"> • Log severity. Sets the level of logging to Errors Only (default), or Debug. The severity setting applies for all the logs that you enable below. You should always use the Errors Only log unless specifically instructed to do otherwise by HPE support, since detailed logging may significantly increase the recording time. • Instrumentation Log. Logs messages related to the instrumentation process. Default value: enabled. • Recording Log. Logs messages issued during recording. Default value: enabled. • Code Generation Log. Logs messages issued during the code generation stage. Default value: enabled.
Remote Objects	For information about this property, see "Remote Objects Property" below .
Serialization	<ul style="list-style-type: none"> • Serialization format. The format of the serialization file that VuGen creates while recording a class that supports serialization: Binary, XML, or Both. The advantage of the binary format is that it is more compressed. The advantage of the XML format is that you can manipulate the data. • Serialize long arrays. For long arrays containing serializable objects (for example, an array of primitives), use VuGen's serialization mechanism. Enabling this option generates LrReplayUtils.GetSerializedObject calls if the array size is equal to or larger than the Threshold value for long array size. • Threshold value for long array size. The minimum size for an array to be serialized if Serialize long arrays is set. <div>  <p>Tip: For XML serialization, you can view the content of the XML file. To view the file, select View XML from the right-click menu.</p> </div>

Remote Objects Property

User interface elements are described below:

UI Element	Description
Record in-process objects	<p>Records activity between the client and server when the server is hosted in the same process as the client. Since the actions are not true client/server traffic, it is usually not of interest. When in-process methods are relevant, for example, in certain Enterprise Service applications, you can enable this option to capture them.</p> <p>Default value: disabled.</p>

Asynchronous calls	<p>Specifies how VuGen should handle asynchronous calls on remote objects and their callback methods</p> <ul style="list-style-type: none"> • Call original callbacks by default. Uses the recorded application's original callback when generating and replaying the script. If the callback method is explicitly excluded by a filter, the callback will be excluded even if you enable this option. • Generate asynchronous callbacks. This option defines how VuGen will handle callbacks when the original callbacks are not recorded. <p>For more information, see "Asynchronous Calls" on page 580.</p>
WCF duplex binding	<ul style="list-style-type: none"> • Generate dummy callback handler. Replaces the original callback in duplex communication with a dummy callback, performing the following actions: <ul style="list-style-type: none"> • Store arguments. When the server calls the handler during replay, it saves the method arguments to a key-value in memory map. • Synchronize replay. It stops the script execution until the next response arrives. VuGen places the synchronization at the point that the callback occurred during recording. This is represented in the script by a warning: • Generate unique client base address. If your application employs dual HTTP Binding, since HTTP is inherently not a duplex protocol, the framework uses a standard port to receive response data being passed to the callback. When you attempt to run multiple instances of your application, you may be unable to do so using the same port number. This option replaces the original client base address's port number with a unique port. <p>For background information about WCF duplex binding, see "Recording WCF Duplex Communication" on page 576.</p>




Network > Mapping and Filtering Recording Options

Enables you to set the port mapping and traffic filtering for the recording or code generation. This option lets you include or exclude specific IPs or ports for your Vusers.

To access	Record > Recording Options > Network > Mapping and Filtering
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
------------	-------------

Capture level	<p>For port mapping only: The level of data to capture (relevant only for HTTP based protocols):</p> <ul style="list-style-type: none"> • Socket level data. Capture data using trapping on the socket level only. Port mappings apply in this case (default). • WinINet level data. Capture data using hooks on the WinINet.dll API used by certain HTTP applications. The most common application that uses these hooks is Internet Explorer. Port mappings are not relevant for this level. • Socket level and WinINet level data. Captures data using both mechanisms. WinINet level sends information for applications that use WinINet.dll. Socket level sends data only if it determines that it did not originate from WinINet.dll. Port mapping applies to data that did not originate from WinINet.dll.
Network-level server address mappings for	<p>For port mapping only: Specifies the mappings per protocol. For example, to show only the FTP mappings, select FTP.</p>
<Port mapping list>	<p>A list of the port mappings.</p> <p>You can temporarily disable the settings for an entry by clearing the check box adjacent to it. When the check box is cleared, VuGen ignores the custom settings for the entry and uses the default settings—it does not cause the host:port to be ignored. To filter a host and port, use Traffic Filtering.</p>
<Traffic filtering list>	<p>A list of the traffic filters, indicating the server name, port and filtering level.</p> <ul style="list-style-type: none"> • Click New Entry or Edit Entry to set these values. • Clear the check box adjacent to an entry to disable it temporarily. • When you launch VuGen, all unchecked port mapping entries are converted into traffic filter entities.
	<p>In the Port mapping section: Opens the Server Entry dialog box, allowing you to add a new mapping. For user interface details, see "Server Entry - Port Mapping Dialog Box" on the next page.</p> <p>In the Traffic filtering section: Opens the New Entry dialog box, allowing you to add a new traffic filter. For user interface details, see "Server Entry - Traffic Filtering Dialog Box" on page 190.</p>
	<p>Opens the Server Entry or New Entry dialog box, allowing you to edit the selected entry.</p>
	<p>For port mapping, this button opens the Advanced Settings dialog box to enable auto-detection of the communication protocol and SSL level. For details, see "Advanced Port Mapping Settings Dialog Box" on page 189.</p>

Note:

- In LoadRunner versions prior to 12.02, only port mapping was available, but not traffic filtering.
- If you upgrade from a version of LoadRunner prior to 12.02, the first time you open VuGen, it will automatically convert all of the unchecked port mapping entries into traffic filters.

See also:

- [Port Mapping and Traffic Filtering Overview](#)
- A [blog post](#) that discusses the benefits of port mapping.

Server Entry - Port Mapping Dialog Box


Enables you to define a server from the server list in the network port mapping node.

To access	Record > Recording Options > Network: Mapping and Filtering > Port Mapping > New Entry / Edit Entry
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see " Protocol Compatibility Table " on page 204.

User interface elements are described below:

Section	UI Element	Description
Socket Service	Target Server	The IP address or host name of the target server for which this entry applies. Default value: Any Server.

	Port	<p>The port of the target server for which this entry applies. Entering 0 specifies all ports.</p> <p>If you do not specify all of the port and server names, VuGen uses the following priorities in assigning data to a service:</p> <ul style="list-style-type: none"> • Priority 1: port and server specified • Priority 2: port not specified, server specified • Priority 3: port specified, server not specified • Priority 4: port and server not specified <p>A map entry with a high priority does not get overridden by an entry with a lower priority. For example, if you specify that traffic on server twilight using port 25 be handled as SMTP and then you specify that all servers on port 25 be handled as HTTP, the data will be treated as SMTP.</p> <ul style="list-style-type: none"> • Forced mapping. If you specify a mapping for a port number, server name, or combination server:port, VuGen forces the network traffic to use that service. For example, if you were to specify <Any> server on port 80 to use FTP, VuGen uses the FTP protocol to record that communication, even though the actual communication may be HTTP. In this instance, the Vuser script might be empty.
	Service ID	A protocol or service name used by the recorder to identify the type of connection (i.e. HTTP, FTP, and so on). You can also specify a new name. The name may not exceed 8 characters.
	Service Type	The type of service, currently set to TCP.
	Record Type	The type of recording—directly or through a proxy server.
	Connection Type	The security level of the connection: Plain (non-secure), SSL , or Auto . If you select Auto, the recorder checks the first 4 bytes for an SSL signature. If it detects the SSL signature, it assumes that SSL is being used.
SSL Configuration	SSL Version	<p>The preferred SSL version to use when communicating with the client application and the server.</p> <p>Default value: SSL 2/3. However some services require SSL 3.0 only or SSL 2.0 only. Some new wireless applications require the Transport Layer Security algorithm TLS 1.x, or ALPN.</p>

	SSL Ciphers	<p>The SSL cipher to use when connecting with a remote secure server.</p> <ul style="list-style-type: none"> • To enable OpenSSL to choose the default cipher automatically, select (Default OpenSSL Ciphers) in the dropdown list . • To define your own cipher and add it to the dropdown list, select (New SSL Cipher) and type in the cipher string. • To remove a user-defined cipher, select the cipher in the list and click the  button. <p>For more information on OpenSSL cipher strings, refer to the Cipher Strings section in https://www.openssl.org/docs/manmaster/apps/ciphers.html</p>
	Use specified client-side certificate	The default client-side certificate to use when connecting to a remote server. Specify or browse for a certificate file in txt , crt , or pem format, and supply a password.
	Use specified proxy-server certificate	The default server certificate to present to client applications that request a server certificate. Specify or browse for a certificate file in txt , crt , or pem format, and supply a password. Click Test SSL to check the authentication information against the server.


Advanced Port Mapping Settings Dialog Box

Enables you to set the advanced port mapping settings. For more information, see "[Port Mapping Auto Detection](#)" on page 208.

To access	Record > Recording Options > Network > Mapping and Filtering. Click Options in the Port Mapping section.
Important information	This dialog box is available only for specific protocols. For a complete list of protocols and their associated nodes, see " Protocol Compatibility Table " on page 204.

User interface elements are described below:

UI Element	Description
------------	-------------

Enable auto SSL detection	<p>Automatically detects SSL communication. Note that this applies only to port mappings that were defined as auto in the Connection type box, or not defined at all. If a server, port, or server:port combination was defined as either Plain or SSL, then auto SSL detection does not apply.</p> <p>SSL Version: Specify the version that you want to detect.</p> <p>SSL Ciphers: Specify the cipher to detect:</p> <ul style="list-style-type: none"> To enable OpenSSL to choose the default cipher automatically, select (Default OpenSSL Ciphers) in the dropdown list . To define your own cipher and add it to the dropdown list, select (New SSL Cipher) and type in the cipher string. To remove a user-defined cipher, select the cipher in the list and click the  button. <p>For more information on OpenSSL cipher strings, refer to the Cipher Strings section in https://www.openssl.org/docs/manmaster/apps/ciphers.html</p>
Enable auto detection of SOCKET based communication	<p>Automatically detects the type of communication. If required, raise the maximum number of transitions, one at a time until VuGen succeeds in detecting the protocol. You can also gradually increase the maximum buffer size by 1024 bytes (1 KB) at a time until VuGen succeeds in detecting the protocol. This allows VuGen to review a larger amount of data in order to find a signature.</p>
Log Level	Sets the logging level for the automatic socket detection.

Server Entry - Traffic Filtering Dialog Box

Enables you to define a new entry for traffic filtering.

To access	Record > Recording Options > Network > Mapping and Filtering > New Entry / Edit Entry in the Traffic filtering section.
------------------	---

User interface elements are described below:

UI Element	Description
Target server	<p>The IP address or host name of the target server for which this entry applies. You can use an asterisk (*) as a wildcard to include, for example, multiple servers in a single domain.</p> <p>Default value: All Servers.</p>
Regular Expression	<p>Indicates that the Target server string is a regular expression. You can use a regular expression to define an "all but one" filter. For example, you can filter out all servers that do not contain "acme.com" with the following expression: <code>^(?!.*\acme\.com)</code>.</p>

Port	The ports to which the filtering should be applied: All ports, Specific port, or Port range.
Filtering level	Where to apply the filtering: <ul style="list-style-type: none"> • Recording. Filter the selected entries when during recording. • Code generation. Filter the selected entries during code generation only. If you enable this option, you will be able to retrieve excluded traffic at a later stage, by modifying the traffic filters before regenerating a script. For details, select Record > Regenerate Script and click Options.

RDP > Code Generation > Advanced Recording Options

Enables you to control the way VuGen creates an RDP script. Only advanced users are advised to modify these settings.

To access	Record > Recording Options > RDP > Code Generation - Adv
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Correlate clipboard parameters	Replaces the recorded clipboard text sent by the user with the correlated parameter containing the same text as received from the server.
Double-click timeout (msec)	The maximum time (in milliseconds) between two consecutive mouse button clicks to be considered a double-click. Default value: 500 milliseconds.
Prefix for clipboard parameters	The prefix for clipboard parameters generated in the current script. This is useful when merging scripts, allowing you to specify a different prefix for each script. Default value: ClipboardDataParam_.
Prefix for snapshot names	The prefix for snapshot file names generated in the current script. This is useful when merging scripts—you can specify a different prefix for each script. Default value: snapshot_.

RDP > Code Generation > Agent Recording Options

Enables you to control the way the agent for Microsoft Agent for Terminal Server functions with VuGen during recording.

To access	Record > Recording Options > RDP > Code Generation - Agent
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Use RDP agent	<p>Generates script using extended information gathered by the RDP agent. The RDP agent must be installed on the server. For details, see "Install/Uninstall the RDP Agent" on page 614.</p> <div> <p>Note: To utilize this feature, you must enable the Replay with RDP agent runtime settings. For details, see RDP > RDP Agent view in the runtime settings.</p> </div>
Enable RDP agent log	<p>Enables the RDP agent log.</p> <ul style="list-style-type: none"> • RDP agent log detail level. Configures the level of detail generated in the RDP agent log with Standard being the lowest level of detail and Extended Debug being the highest level of detail. • RDP agent log destination. Configures the destination of the RDP agent log data. File saves the log messages only on the remote server side. Stream sends the log messages to the VuGen machine. FileAndStream sends the log messages to both destinations. • RDP agent log folder. The folder path on the remote server that the RDP agent log file will be generated in.

RDP > Code Generation > Basic Recording Options

Enables you to control the way VuGen creates a script—the level of detail, triggers, and timeouts.

To access	Record > Recording Options > RDP > Code Generation - Basic
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Always generate connection name	<p>If selected, function call will contain the ConnectionName parameter. If not selected, the functions will only contain this parameter if more than a single rdp_connect_server appears in the script.</p> <p>Default value: disabled.</p>

UI Element	Description
Automatic generation of synchronization points	<p>Synchronization points allow the script to pause in the replay while waiting for a window or dialog to pop-up, or some other control to fulfil a certain condition. This option automatically generates sync_on_image functions before mouse clicks and drags (enabled by default). The Sync radius is the distance from the mouse operation to the sides of the rectangle which defines the synchronization area. The default is 20 pixels. Select one of the following options:</p> <ul style="list-style-type: none"> • None. No synchronization points are automatically added. • Rectangular. Creates synchronization points as rectangular boxes centered around the click or drag location. • Enhanced. Creates synchronization points designed to select only a particular location and to react to changes in the UI at that location. For example, a button may be selected and a synchronization point created if the button moves. If a synchronization region is not recognized, the rectangular synchronization settings are used.
Generate mouse movement calls	<p>Generates rdp_mouse_move calls in the script. When enabled, this option significantly increases the script size.</p> <p>Default value: disabled.</p>
Generate raw keyboard calls	<p>Generates rdp_raw_key_up/down calls as if the script level was set to Raw. Mouse calls will still be generated according to the script level. If disabled, VuGen generates Keyboard calls according to the script level. If the script level is set to Raw, this option is ignored.</p> <p>Default value: disabled.</p>
Generate raw mouse calls	<p>Generates rdp_mouse_button_up/down calls as if the script level was set to Raw. Keyboard calls will still be generated according to the script level. If disabled, VuGen generates Mouse calls according to the script level. If the script level is set to Raw, this option is ignored.</p> <p>Default value: disabled.</p>

UI Element	Description
Script generation level	<p>The level of the script and the type of API functions to use when generating the script.</p> <ul style="list-style-type: none"> • High. Generate high level scripts. Keyboard events are translated to rdp_type calls. Two consecutive mouse clicks with the same coordinates are translated as a double-click. • Low. Generate low level scripts. Key up/down events are translated into rdp_key events. Modifier keys (Alt, Ctrl, Shift) are used as a KeyModifier parameter for other functions. Mouse up/down/ move events are translated to mouse click/drag events. • Raw. Generates a script on a raw level, by extracting input events from network buffers and generating calls in their simplest form: key up/down, mouse up/down/move. The KeyModifier parameter is not used.

RDP > Client Startup Recording Options

Enables you to set the RDP client startup recording options.

To access	Record > Recording Options > RDP > Client Startup
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Run RDP client application	Connects to the terminal server by running the Terminal Services client.
Use custom connection file	Connects to the terminal server by using an existing connection file. The file should have an *.rdp extension. You can browse for the file on your file system or network.
Use default connection file	Connects to the terminal server by using the Default.rdp file in your document's folder.

Recording Properties > Corba Options Recording Options

Enables you to set the CORBA specific recording properties and several callback options.

To access	Record > Recording Options > Recording Properties > Corba Options
------------------	---

Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .
------------------------------	---

User interface elements are described below:

UI Element	Description
Record Callback Connection	Instructs VuGen to generate a connect statement for the connection to the ORB, for each callback object. Default value: disabled.
Record DLL only	Instructs VuGen to record only on a DLL level. Default value: disabled.
Record Properties	Instructs VuGen to record system and custom properties related to the protocol. Default value: enabled.
Resolve CORBA Objects	When correlation fails to resolve a CORBA object, recreate it using its binary data. Default value: disabled.
Show IDL Constructs	Displays the IDL construct that is used when passed as a parameter to a CORBA invocation. Default value: enabled.
Use local vendor classes	Use local vendor classes and add the srv folder to the BOOT classpath. If you disable this option, VuGen uses network classes and adds the script's classes to the classpath. Default value: enabled.
Vendor	The CORBA vendors: Inprise Visibroker , Iona OrbixWeb , or Bea Weblogic .

Recording Properties > Correlation Options - Recording Options

Allows you to enable automatic correlation, and control its depth.

To access	Record > Recording Options > Recording Properties > Correlation Options
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Advanced Correlation	Enables correlation on complex objects such as arrays and CORBA container constructs and arrays. This type of correlation is also known as deep correlation. Default value: enabled.


Correlate Collection Type	Correlates objects from the Collection class for JDK 1.2 and higher. Default value: disabled.
Correlate String Arrays	Correlate strings within string arrays during recording. If disabled, strings within arrays are not correlated and the actual values are placed in the script. Default value: enabled.
Correlate Strings	Correlate strings in script during recording. If disabled, the actual recorded values are included in the script between quotation marks and all other correlation options are ignored Default value: disabled.
Correlation Level	Indicates the level of deep correlation, the number of inner containers to be scanned. Default value: 15.


Recording Properties > Log Options Recording Options

Enables you to determine the level of debug information generated during recording.

To access	Record > Recording Options > Recording Properties > Log Options
Important information	<p>This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204.</p> <p>These options are not supported for the Java Record Replay protocol when using Java 8.</p>

User interface elements are described below:

UI Element	Description
Class Dumping	<p>Dumps all of the loaded classes to the script folder. Default value: disabled.</p> <div>  <p>Tip: Under the script's data folder, VuGen creates a subfolder named dump. This folder will contain a copy of each class file that was loaded. You can use these class files to determine the signatures when defining custom hooks.</p> </div>
Digest Calculation	<p>Generate a digest of all recorded objects. Default value: disabled.</p> <ul style="list-style-type: none"> • Exclude from Digest. A list of objects not to be included in the digest calculation. <div> <p>Syntax: java.lang.Object class format, delimiter = ","</p> </div>

Log Level	<p>The level of recording log to generate:</p> <ul style="list-style-type: none"> • None. No log file is created • Brief. Generates a standard recording log and output redirection • Detailed. Generates a detailed log for methods, arguments, and return values. • Debug. Records hooking and recording debug information, along with all of the above. <div>  Note: The log files will be stored in the script folder's data directory. </div>
Synchronize Threads	<p>For multi-threaded applications, instructs VuGen to synchronize between the different threads. Default value: disabled.</p>

Recording Properties > Recorder Options - Recording Options

Enables you to set the Java protocol to record as well as other protocol specific recording options.

To access	Record > Recording Options > Recording Properties > Recorder Options
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Byte Array Format	<p>The format of byte arrays in a script: Regular, Unfolded Serialized Objects, or Folded Serialized Objects. Use one of the serialized object options when recording very long byte arrays. Default value: Regular.</p>
Bytes as Characters	<p>Displays readable characters as characters with the necessary casting—not in byte or hexadecimal form. Default value: enabled.</p>
Comment Lines Containing	<p>Comment out all lines in the script containing one of the specified strings. To specify multiple strings, separate the entries with commas. Default value: Any line with a string containing <undefined> will be commented out.</p>
Extensions List	<p>A comma separated list of all supported extensions. Each extension has its own hooks file. Default value: JNDI.</p>

UI Element	Description
Insert Functional Check	Inserts verification code that compares the return value received during replay, to the expected return value generated during recording. This option only applies to primitive return values. Default value: disabled.
Load Parent Class Before Class	Change the loading order so that parent classed are loaded before child classes. This helps identify hooking for trees with deep inheritance. Default value: enabled.
Record LoadRunner Callback	Records the LoadRunner stub object as a callback. If disabled, VuGen records the original class as the callback. Default value: enabled.
Recorded Protocol	Specifies which protocol to record, For supported protocols, see "Supported Java Communication Protocols" on page 516 . Default value: RMI.
Remove Lines Containing	Remove all lines containing one of the specified strings from the script. To specify multiple strings, separate the entries with commas. This feature is useful for customizing the script for a specific testing goal.
Unreadable Strings as Bytes	Represents strings containing unreadable characters as byte arrays. This option applies to strings that are passed as parameters to invocations. Default value: enabled.
Use _JAVA_OPTIONS flag	Forces JVM versions 1.2 and higher to use the _JAVA_OPTION environment variable which contains the desired JVM parameters. Default value: disabled.
Use DLL hooking to attach LoadRunner support	Use DLL hooking to automatically attach LoadRunner support to any JVM.

Recording Properties > Serialization Options - Recording Options

Enables you to control how objects are serialized. Serialization is often relevant to displaying objects in an ASCII representation in order to parameterize their values.

To access	Record > Recording Options > Recording Properties > Serialization Options
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, "Protocol Compatibility Table" on page 204 .
See also	How to Correlate Scripts - Java Scripts - Serialization

User interface elements are described below:

UI Element	Description
Unfold Serialized Objects	<p>Expands serialized objects in ASCII representation and allows you to view the ASCII values of the objects in order to perform parameterization.</p> <ul style="list-style-type: none"> • Limit Object Size (bytes). Limits serializable objects to the specified value. Objects whose size exceeds this value, will not be given ASCII representation in the script. Default value: 3072 bytes. • Ignore Serialized Objects. Lists the serialized objects not to be unfolded when encountered in the recorded script. Separate objects with commas. Syntax: java.lang.Object class format, delimiter = "," • Serialization Delimiter. Indicates the delimiter separating the elements in the ASCII representation of objects. VuGen will only parameterize strings contained within these delimiters. The default delimiter is `#`. • Unfold Arrays. Expands array elements of serialized objects in ASCII representation. If you disable this option and an object contains an array, the object will not be expanded. Default value: enabled—all deserialized objects are totally unfolded. • Limit Array Entries. Instructs the recorder not to open arrays with more than the specified number of elements. Default value: 200.

RTE > Configuration Recording Options

Enables you to set the recording options to match the character set used during terminal emulation.

To access	Record > Recording Options > RTE > Configuration
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Character Set	Match the character set used during terminal emulation. The default character set is ANSI. For Kanji and other multi-byte platforms, you can specify DBCS (Double-byte Character Set).

RTE > RTE Recording Options

Enables you to set the general RTE recording options.

To access	Record > Recording Options > RTE > RTE
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Generate automatic synchronization commands	<p>Automatically generates a number of TE-synchronization functions, and insert them into the script while you record.</p> <ul style="list-style-type: none"> • Cursor. Generate a TE_wait_cursor function before each TE_type function. • Prompt. Generate a TE_wait_text function before each TE_type function (where appropriate). • X-System. Generate a TE_wait_sync function each time a new screen is displayed while recording. <p>Note: VuGen generates meaningful TE_wait_text functions when recording VT type terminals only. Do not use automatic TE_wait_text function generation when recording block-mode (IBM) terminals.</p>
Generate automatic X-System transaction	<p>Records the time that the system was in the X SYSTEM mode during a scenario run. This is accomplished by inserting a TE_wait_sync_transaction function after each TE_wait_sync function. Each TE_wait_sync_transaction function creates a transaction with the name default. Each TE_wait_sync_transaction function records the time that the system spent in the previous X SYSTEM state.</p>
Generate screen header comments	<p>Generates screen header comments while recording a Vuser script, and inserts the comments into the script. A generated comment contains the text that appears on the first line of the terminal emulator window.</p> <p>Note: You can generate comments automatically only when using block-mode terminal emulators such as the IBM 5250.</p>
Keyboard record timeout	<p>When you type text into a terminal emulator while recording, VuGen monitors the text input. After each keystroke, VuGen waits up to a specified amount of time for the next key stroke. If there is no subsequent keystroke within the specified time, VuGen assumes that the command is complete.</p>

SAPGUI > Auto Logon Recording Options

Enables you to log on automatically when you begin recording. The logon functions are placed in the `vuser_init` section of the script.

To access	Record > Recording Options > SAPGUI > Auto Logon
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Enable Auto logon	Enables you to log on automatically when you begin recording. Enter the Server name , User , Password , Client name, and interface Languages for the SAP server.

SAPGUI > Code Generation Recording Options

Enables you to set the code generation settings for the SAPGUI protocol.

To access	Record > Recording Options > SAPGUI > Code Generation
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:

UI Element	Description
Always generate Object ID in header file	Places the Object IDs in a separate header file instead of in the script. When you disable this option, VuGen generates the IDs according to the specified string length in the general script setting. This results in a more compact and cleaner script.
Generate Fill Data steps	Generates Fill Data steps for table and grid controls—instead of separate steps for each cell.
Generate logon operation as a single step	Generates a single sapgui_logon method for all of the logon operations. This helps simplify the code. If you encounter login problems, disable this option.

SAPGUI > General Recording Options

Enables you to set the general recording options for the SAPGUI protocol.

To access	Record > Recording Options > SAPGUI > General
Important information	This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see "Protocol Compatibility Table" on page 204 .

User interface elements are described below:



UI Element	Description
Capture screen snapshots	Indicates how to save the snapshots of the SAPGUI screens as they appear during recording: ActiveScreensnapshots , Regular snapshots , or None . ActiveScreen snapshots provide more interactivity and screen information after recording, but they require more resources.
Changing events during recording	Process Context menus by text. Processes context menus by their text, generating sapgui_toolbar_select_context_menu_item_by_text functions. When disabled, VuGen processes context menus by their IDs, and generates a sapgui_toolbar_select_context_menu_item for context menus. This is an advantage when working with Japanese characters.



Traffic Analysis > Traffic Filters Recording Options

This dialog box enables you to filter either incoming or outgoing traffic.

To access	Record > Recording Options > Traffic Analysis > Traffic Filters
Important information	<ul style="list-style-type: none"> For details, see Analyzing Traffic. This node is available only for specific protocols. For a complete list of protocols and their associated nodes, see the "Protocol Compatibility Table" on page 204.
Relevant tasks	"Create a Vuser Script by Analyzing a Captured Traffic File" on page 685

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
When generating the script	<p>Include all IP addressing the list. Includes traffic from specified IP address.</p> <p>Exclude all IP address in the list. Excludes traffic from specified IP addresses.</p>
<Incoming Traffic Tab> Enables you to identify IP addresses for incoming traffic	
	Adds an IP address.
Source IP	The IP address of the server.
	Deletes an IP address.



UI Element	Description
<Outgoing Traffic Tab> Enables you to identify IP addresses for outgoing traffic	
	Adds an IP address.
Destination IP	The IP address of the server.
Destination Port	The destination port of the server.
	Deletes an IP address.

WinSock Recording Options

Enables you to set the WinSock recording options.

To access	Record > Recording Options > WinSock
Important information	When using translation tables on Solaris machines, you must set the following environment variables on all machines running scripts: <pre>setenv LRSDRV_SERVER_FORMAT 0025 setenv LRSDRV_CLIENT_FORMAT 04e4</pre>
Relevant tasks	"Record a Windows Sockets Script" on page 824

User interface elements are described below:

UI Element	Description
	Adds a new entry to the list of excluded sockets.
	Removes the selected entry from the list of excluded sockets.
Do not include excluded socket in log	Excludes the sockets on the list from the log. Clearing this option enables logging for the excluded sockets. Their actions are preceded by "Exclude" in the log file.

Exclude Settings/Socket List	<p>The host and port of the sockets to exclude from the recording or regeneration of the script. Use the following syntax:</p> <ul style="list-style-type: none"> • host:port format excludes a specific port. • host format excludes all ports for the specified host. • :port format excludes a specific port on the local host. • *:port format excludes a specific port on all hosts.
Encoding Method	<p>Use OEM encoding. Enable data encoding that supports non-English characters.</p> <p>Use ASCII encoding. Enable data encoding that is limited to English characters. Use this option to replicate the LR 9.5x data encoding method.</p>
Think Time Threshold	<p>During recording, VuGen automatically inserts the think time steps when you pause between actions. You can set a threshold level, below which the recorded think time will be ignored.</p> <p>Default value: five seconds.</p>
Translation tables	<p>The Translation Table lets you specify the format for recording sessions when using the WinSock single protocol, and for code generation when using a WinSock multi protocol. This applies to users running on mainframe machines or AS/400 servers. Both the server and client machines determine the format of the data from translation tables installed on your system. Select a translation option from the list box. The first four digits of the listbox item represent the server format. The last four digits represent the client format.</p> <p>The translation tables are located in the ebcdic folder under the VuGen's installation folder. If your system uses different translation tables, copy them to the ebcdic folder.</p> <div> <p>Note: If your data is in ASCII format, it does not require translation. Select the None option, the default value.</p> </div>

Recording Options - Miscellaneous Topics


This section contains a variety of topics relating to recording options.

Protocol Compatibility Table

The following table lists the Vuser protocols and which recording option nodes are available for each protocol.

Protocol	Recording Options Nodes
.NET	<ul style="list-style-type: none"> • General - Script • Microsoft .NET - Recording, Shared DLLs

Protocol	Recording Options Nodes
Ajax - Click and Script	<ul style="list-style-type: none"> General - Recording, Script GUI Properties - Advanced, Web Event Configuration HTTP Properties - Advanced Network - Mapping and Filtering Correlations - Rules
TruClient	<ul style="list-style-type: none"> None
C Vuser	<ul style="list-style-type: none"> None
Citrix ICA	<ul style="list-style-type: none"> General - Script Citrix - Configuration, Recorder, Code Generation, Login <p>Note: The Citrix Login node is available only when creating a single protocol Citrix script. It is not available when creating a multi-protocol Citrix+Web script.</p>
COM/DCOM	<ul style="list-style-type: none"> General - Script COM/DCOM - Filter, Options
DNS	<ul style="list-style-type: none"> None
FTP	<ul style="list-style-type: none"> General - Script Network - Mapping and Filtering
Flex	<ul style="list-style-type: none"> General - Recording, Script, Protocols, Code Generation Flex - RTMP, Configuration, Externalizable Objects Correlations - Configuration, Rules HTTP Properties - Advanced Network - Mapping and Filtering Data Format Extension - Chain Configuration, Code Generation
IMAP	<ul style="list-style-type: none"> General - Script Network - Mapping and Filtering
Java over HTTP	<ul style="list-style-type: none"> General - Recording Correlations - Configuration, Rules HTTP Properties - Advanced Network - Mapping and Filtering Java Environment Settings - Java VM, Classpath Data Format Extension - Chain Configuration, Code Generation

Protocol	Recording Options Nodes
Java Record Replay	<ul style="list-style-type: none"> Java Environment Settings - Java VM, Classpath Recording Properties - Recorder Options, Serialization Options, Correlation Options, Log Options, Corba Options <div>  Note: Log Options are not supported when using Java 8. </div>
Java Vuser	<ul style="list-style-type: none"> None
LDAP	<ul style="list-style-type: none"> General - Script
MAPI (Microsoft Exchange)	<ul style="list-style-type: none"> None
MQTT	<ul style="list-style-type: none"> None
ODBC	<ul style="list-style-type: none"> General - Script Database - Database
Oracle - 2-Tier	<ul style="list-style-type: none"> General - Script Database - Database
Oracle NCA	<ul style="list-style-type: none"> General - Script, Protocols, Recording HTTP Properties - Advanced Correlations - Configuration, Rules Network - Mapping and Filtering
Oracle - Web	<ul style="list-style-type: none"> General - Script, Recording Correlations - Configurations, Rules HTTP Properties - Advanced Network - Mapping and Filtering Data Format Extension - Chain Configuration, Code Generation
POP3	<ul style="list-style-type: none"> General - Script Network - Port Mapping
RDP (Remote Desktop Protocol)	<ul style="list-style-type: none"> General - Script RDP - Client Startup, Code Generation (Basic, Advanced, and Agent) Network - Mapping and Filtering

Protocol	Recording Options Nodes
SAP - Web	<ul style="list-style-type: none"> • General - Recording, Script, Protocols, Code Generation • Correlations - Configuration, Rules • HTTP Properties - Advanced • Network - Mapping and Filtering
SAP GUI	<ul style="list-style-type: none"> • General - Script • SAP GUI - General, Code Generation, Auto Logon
Siebel - Web	<ul style="list-style-type: none"> • General - Recording, Script, Protocols • HTTP Properties - Advanced • Network - Mapping and Filtering • Correlations - Rules
SMTP	<ul style="list-style-type: none"> • General - Script • Network - Mapping and Filtering
Web - HTTP/HTML	<ul style="list-style-type: none"> • General - Recording, Script, Protocols, Code Generation • Correlations - Configuration, Rules • HTTP Properties - Advanced • Network - Mapping and Filtering • Data Format Extension - Chain Configuration, Code Generation
Web Services	<ul style="list-style-type: none"> • General - Recording, Script, Protocols, Code Generation • Correlation - Configuration, Rules • HTTP Properties - Advanced • Traffic Analysis - Traffic Filters • Network - Mapping and Filtering
Windows Sockets	<ul style="list-style-type: none"> • Sockets - Winsock

Port Mapping and Traffic Filtering Overview

When you record a business process, a portion of the generated traffic is not related to the actual business process. For example, the Chrome browser accesses many external servers. This overhead may not be meaningful for the load test.

In addition, as a tester, you may not be interested in some of the generated traffic, even if you generate it during a recording session.

Another issue is a non-Internet business process. If you do not have Internet access, VuGen can successfully record the business process, but it will fail if you use a browser that constantly attempts to access the Internet.

The Port Mapping and Traffic Filtering features allow you to specify the behavior of specific traffic or exclude certain server:port combinations from your Vuser script.

Port Mapping

When recording Vuser scripts that record network traffic on a socket level (HTTP, SMTP, POP3, FTP, IMAP, Oracle NCA and WinSock), you can set the Port Mapping options. Using these options, you can map the traffic from a specific server:port combination to the desired communication protocol.

The available communication protocols to which you can map are FTP, HTTP, IMAP, NCA, POP3, SMTP, and SOCKET. You create a mapping by specifying a server name, port number, or a complete server:port combination. For example, you can indicate that all traffic from the server *twilight* on port 25, should be handled as SMTP. You can also specify that all traffic from the server called *viper*, should be mapped to the FTP protocol, regardless of the port. Additionally, you can map all traffic on port 23 to SMTP, regardless of the server name.

When recording in multi-protocol mode, If at least one of the protocols records on a socket level, the **Mapping and Filtering** node will be available. Wildcards in the server name are not supported for port mapping.

For details on adding new port mappings, see ["Server Entry - Port Mapping Dialog Box" on page 187](#).

Traffic Filtering

Traffic filtering extends the capabilities of port mapping by letting you list URLs and ports to exclude. In port mapping, you cannot use wildcards.

Using traffic filtering, you add an entry for each server that you want to exclude. You can use wildcards to exclude all traffic associated with a specific domain.

You may also specify a port or a range of ports. For example, you can filter out only SSL traffic coming through port 443. Once you define an entry, you can clear its check box to temporarily disable it.

You can select a filtering level:

- Recording
- Code generation

The advantage of excluding undesired traffic from the recording session, is that your script will be lighter, increasing its performance.

The benefit of only excluding traffic from the code generation, is that traffic will be recorded and will be accessible to you if you need it at a later point. You can then reapply a different filter without having to rerecord the business process.

For details on adding new traffic filters, see the ["Server Entry - Traffic Filtering Dialog Box" on page 190](#).

Port Mapping Auto Detection

VuGen's advanced Port mapping options let you configure the **auto-detection** options. VuGen's auto-detection analyzes the data that is sent to the server. It checks the data for a signature, a pattern in the data's content, that identifies the protocol. For the purpose of detecting a signature, all of the send buffers until the first receive buffer, are combined. All send buffers that were sent until a receive buffer

is returned, are considered a single data **transition**. By default, no mappings are defined and VuGen employs auto-detection. In some protocols, VuGen determines the type in a single transition, (such as HTTP). Other network protocols require several transitions before determining the type. For this purpose, VuGen creates a temporary buffer for each server-port combination. If VuGen cannot determine the protocol type by reading the first transition buffers, it stores the data in a temporary buffer. It continues to read the incoming buffers until it detects a signature of a specific protocol.

By default, VuGen allows 4 transitions and uses a temporary buffer of 2048 bytes in order to detect a protocol signature. If VuGen has not yet determined the type after reaching the maximum number of transitions, or after reaching the maximum buffer size, it assigns the data to the WinSock protocol. If you did not instruct VuGen to record the WinSock protocol (in the multi-protocol selection), VuGen discards the data.

You can change the maximum number of buffers you want VuGen to read in order to detect the protocol type. You can also specify the size of the temporary buffer. In instances where the amount of data in the first send buffers, is greater than the size of the temporary buffer, VuGen cannot auto-detect the protocol type. In this case, you should increase the size of the temporary buffer.

When working with the above network level protocols, we recommend that you allow VuGen to use auto-detection to determine the protocol type. In most cases, VuGen's recorder is able to recognize the signatures of these protocols. It then automatically processes them according to the protocol specifications. In certain instances, however, VuGen may be unable to recognize the protocol. For example:

- The protocol signature closely resembles an existing protocol, resulting in erroneous processing.
- There is no unique signature for the protocol.
- The protocol uses SSL encryption, and therefore cannot be recognized on a WinSock level.

In all of the above cases, you can supply information to uniquely identify the server and port hosting the protocol.

EUC-Encoding (Japanese Windows only)

When working with non-Windows standard character sets, you may need to perform a code conversion. A character set is a mapping from a set of characters to a set of integers. This mapping forms a unique character-integer combination for a given alphabet. Extended UNIX Code (EUC) and Shift Japan Industry Standard (SJIS) are non-Windows standard character sets used to display Japanese characters on Web sites.

Windows uses SJIS encoding, while UNIX uses EUC encoding. When a Web server is running UNIX and the client is running Windows, the characters in a Web site are not displayed on the client machine properly due to the difference in the encoding methods. This affects the display of EUC-encoded Japanese characters in a Vuser script.

During recording, VuGen detects the encoding of a Web page through its HTTP header. If the information on the character set is not present in the HTTP header, it checks the HTML meta tag.

If you know in advance that a Web page is encoded in EUC, you can instruct VuGen to use the correct encoding by using the recording options. To record a page in EUC-encoding, enable the **EUC** option in the Recording Options **Recording** node (only visible for Japanese Windows).

Enabling the **EUC** option forces VuGen to record a Web page in EUC encoding, even when it is not EUC-encoded. Therefore, you should only enable this option when VuGen cannot detect the encoding from the HTTP header or the HTML meta tag or when you know in advance that the page is EUC-encoded.

During recording, VuGen receives an EUC-encoded string from the Web server and converts it to SJIS. The SJIS string is saved in the script's **Action** function. However, for replay to succeed, the string has to be converted back to EUC before being sent back to the Web server. Therefore, VuGen adds a **web_sjis_to_euc_param** function before the **Action** function, which converts the SJIS string back to EUC.

In the following example, the user navigates to an EUC-encoded Web page and clicks a link. VuGen records the **Action** function and adds the **web_sjis_to_euc_param** function to the script before the **Action** function.

```
web_sjis_to_euc_param("param_link","Search");  
web_link("LinkStep","Text={param_link}");
```

For more information, see ["Advanced URL Dialog Box" on page 167](#).

Script Generation Preference Overview

Before you record a session, VuGen allows you to specify a language for script generation. The available languages for script generation vary per protocol. The most common available languages are C and Java. By default, VuGen generates a script in the most common language for that protocol, but you can change this through the **Script** recording options node.

For user interface details, see ["General > Script Recording Options" on page 169](#).



Tip: If you record a script in one language, you can regenerate it in another language after the recording. For task details, see ["Regenerate a Vuser Script" on page 220](#).

After you select a generation language, you can enable language-specific recording options which instruct the recorder what to include in the script and how to generate it.

If at least one of the protocols you are recording has multi-protocol capabilities, the Script node will be available except when you record HTTP or WinSock as a single protocol script.

Script Language Options

When you record a session, VuGen creates a script that emulates your actions. The default script generation language is C. The following list specifies which protocols are appropriate for each language:

- **C.** For recording applications that use complex COM constructs and C++ objects.
- **C #.** For recording applications that use complex applications and environments (MS .NET protocol only).
- **Visual Basic .NET.** For VB .NET applications using the full capabilities of VB.
- **JavaScript.** For Web-based applications , especially those using dynamic HTML applications.

After the recording session, you can modify the script with regular C, C#, .NET, or JavaScript code and control flow statements.

Recording Levels - Overview

VuGen lets you specify what information to record and which functions to use when generating a Vuser script by selecting a recording level in the **General > Recording** node of the **Recording Options** dialog box.

The recording level you select depends on your needs and environment.

HTML-based script

Generates a separate step for each HTML user action. The steps are intuitive, but they do not reflect true emulation of the JavaScript code.

```
/* HTML-based mode - a script describing user actions*/
...
web_url("WebTours",
    "URL=http://localhost/WebTours/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    LAST);
web_link("Click Here For Additional Restrictions",
    "Text=Click Here For Additional Restrictions",
    "Snapshot=t4.inf",
    LAST);
web_image("buttonhelp.gif",
    "Src=/images/buttonhelp.gif",
    "Snapshot=t5.inf",
    LAST);
...
```

URL-based script

Records all browser requests and resources from the server that were sent due to the user's actions. Automatically records all HTTP resources as URL steps (**web_url** statements). For normal browser recordings, it is not recommended to use the URL-based mode since it is more prone to correlation related issues. However, if you are recording pages such as applets and non-browser applications, this mode is ideal.

URL-based scripts are not as intuitive as the HTML-based scripts since all actions are recorded as **web_url steps** instead of **web_link**, **web_image**, and so on.

```
/* URL-based mode - only web_url functions */
...
web_url("spacer.gif",
    "URL=http://graphics.hplab.com/images/spacer.gif",
    "Resource=1",
    "RecContentType=image/gif",
    "Referer=",
```

```
        "Mode=HTTP",  
        LAST);  
web_url("calendar_functions.js",  
        "URL=http://www.im.hplab.com/travelp/calendar_functions.js",  
        "Resource=1",  
        "RecContentType=application/x-javascript",  
        "Referer=",  
        "Mode=HTTP",  
        LAST);  
...
```

You can switch recording levels and advanced recording options while recording, provided that you are not recording a multi-protocol script. The option of combining recording levels is available to advanced users for performance testing.

You can also regenerate a script after recording, using a different method than the original recording. For example, if you record a script on an HTML-based level, you can regenerate it on a URL-based level. To regenerate a script, select **Record > Regenerate Script** and click **Options** to set the recording options for the regeneration.

See also:

- For user interface details, see ["General > Recording - Recording Options" on page 166](#).

Serialization Overview

VuGen uses serialization when it encounters an unknown object during the recording, provided that the object supports serialization. An unknown object can be an input argument which was not included by the filter and therefore its construction was not recorded. Serialization helps prevent compilation errors caused by the passing of an unknown argument to a method. If an object is serialized, it is often advisable to set a custom filter to record this object. For details, see ["Serialize Flex Scripts" on page 512](#).

Tips for Working with Event Listening and Recording

It can sometimes be difficult to find the ideal listen and recording settings. When defining these settings, keep in mind the following guidelines:

- To record an event on an object, you must instruct VuGen to listen for the event, and to record the event when it occurs. You can listen for an event on a child object, even if a parent object contains the handler or behavior, or you can listen for an event on a parent object, even if the child object contains the handler or behavior.

However, you must enable recording for the event on the source object (the one on which the event actually occurs, regardless of which parent object contains the handler or behavior).

For example, suppose a table cell with an **onmouse over** event handler contains two images. When a user touches either of the images with the mouse pointer, the event bubbles up to the cell and includes information on which image was actually touched. You can record this mouse over event by:

- Setting **Listen** on the WebTable mouse over event to **If Handler** (so that VuGen "hears" the event

when it occurs), while disabling recording on it, and then setting **Listen** on the Image mouse over event to **Never**, while setting its recording status to **Enable** (to record the mouse over event on the image after it is listened to at the WebTable level).

- Setting **Listen** on the Image mouse over event to **Always** (to listen for the mouse over event even though the image tag does not contain a behavior or handler), and setting the recording status on the Image object to **Enabled** (to record the mouse over event on the image).
- Instructing VuGen to listen for many events on many objects may lower performance, so try to limit listening settings to the required objects.
- In rare situations, listening to the object on which the event occurs (the source object) may interfere with the event.

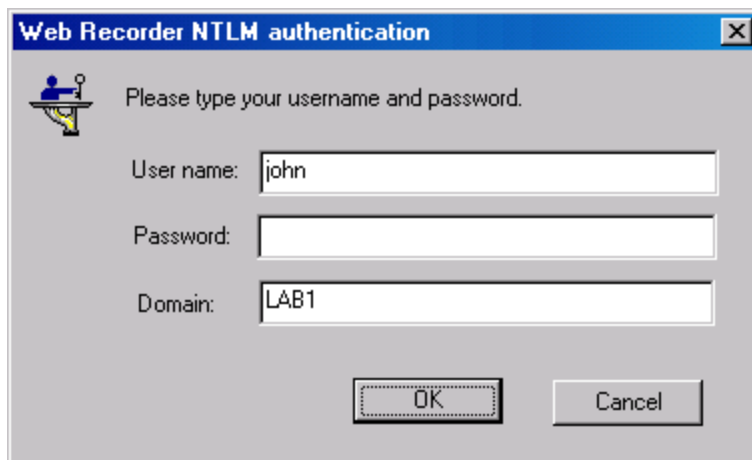
Providing Authentication Information for Multi-Protocol Scripts

When recording a Web session that uses NTLM authentication, your server may require you to enter details such as a user name and password.

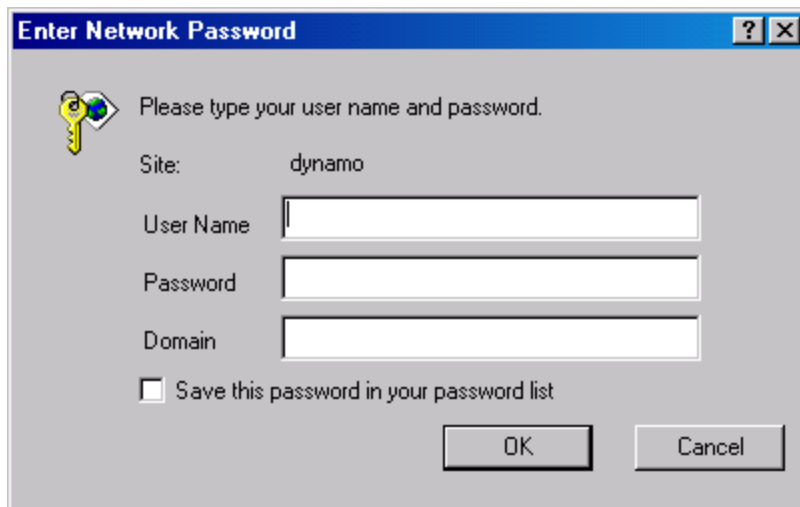
How Internet Explorer authenticates users

Initially, IE (Internet Explorer) tries to use the NT authentication information of the current user:

- If IE succeeds in logging in using this information and you record a script—then, at the end of the recording VuGen prompts you to enter a password. VuGen retrieves the user name and domain information automatically. If necessary, you can also edit the user name in the Web Recorder NTLM authentication dialog box.



- If IE is unable to log in with the current user's information, it prompts you to enter a user name and password using the standard browser authentication dialog box.



Generating a web_set_user function

When performing NTLM authentication, VuGen adds a **web_set_user** function to the script.

- If authentication succeeds, VuGen generates a **web_set_user** function with your user name, masked password, and host.

```
web_set_user("domain1\dashwood",  
            lr_unmask("4042e3e7c8bbbcfd0f737f91f"),  
            "sussex:8080");
```

- If you cancel the Web Recorder NTLM Authentication dialog box without entering information, VuGen generates a **web_set_user** function for you to edit manually.

```
web_set_user("domain1\dashwood",  
            "Enter NTLM Password Here",  
            "sussex:8080");
```

If you enter a password manually, it will appear in the script as-is, presenting a security issue.

To mask a password:

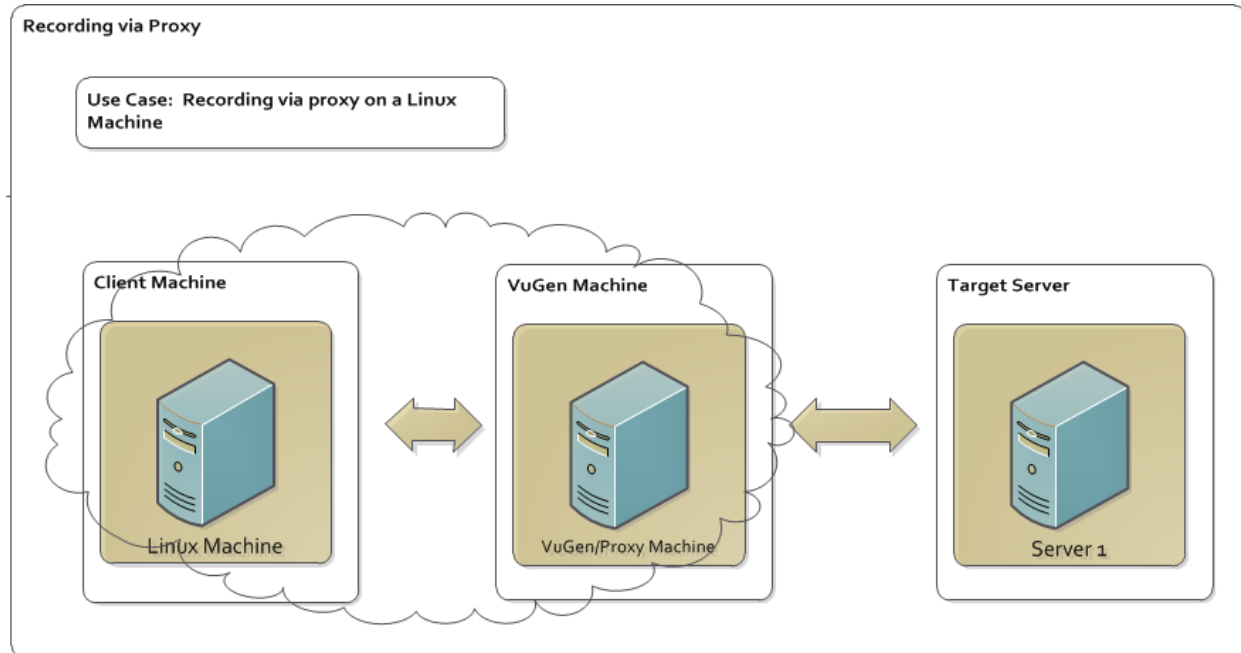
Right-click the password and select **Mask String**. VuGen masks the string and generates an **lr_unmask** function, used to decode the password during replay. For more information about masking strings, see ["Password Encoder" on page 332](#).

Recording via a Proxy - Overview

VuGen allows you to record scripts using a LoadRunner proxy to resolve situations where you cannot install VuGen on the client machine. This may be the case with certain Linux machines, Mac OS machines, and mobile devices.

When using this option, the VuGen machine acts as a proxy server capturing all the traffic from the client machine to the target server. After the business process has been recorded VuGen creates a script.

The following diagram illustrates the basic workflow:



Considerations for Recording via a Proxy:

- **Supported only for some protocols.** Recording via a proxy is not available for all protocols. Partial list of supported protocols: Web - HTTP/HTML, Flex, Java over HTTP, Oracle NCA, and Oracle - Web
- **Allow proxy configurations.** The client must allow proxy configurations, meaning, you must be able to specify the port and the address of the VuGen machine on the client device or machine.
- **Same network.** The client device or machine and the VuGen machine must be in the same network.
- **Delete browsing history.** Make sure to delete the client machine's browsing history prior to recording the business process.
This is because VuGen cannot bypass the browser's cache and history settings on the client machine. Deleting the browsing history enables VuGen to accurately record your business process via a proxy.
- **HTTP forwarding to multiple targets.** If multiple target machines are present, the VuGen proxy can correctly forward data to the right target server according to the Host HTTP header.
- **Forwarding to target server via Internet proxy server** You can configure the VuGen machine to establish a connection with your organization's internet proxy by selecting the **Remote Application via LoadRunner proxy** mode in the **Start Recording** dialog box. For details, see ["Start Recording Dialog Box" on page 221](#).

Record a Script via a Proxy

This topic describes various methods for recording a script using a proxy server. In all use cases, the client machine and VuGen machine are in the same network.

Use Case 1

You want to record a business process but you cannot install VuGen on the client machine or device.

1. Create a new **Web - HTTP/HTML** script.
2. Start Recording.
 - a. From the **Start Recording** dialog box, select **Recording Mode > Record > Remote Application via LoadRunner Proxy**. For details, see ["Start Recording Dialog Box" on page 221](#).
 - b. Specify the port on which the LoadRunner proxy will listen, by default, port 8888.
 - c. Check the **Display recording toolbar on client machine**. This allows you to see and interact with the recording toolbar on the client machine.
 - d. Click **Start Recording**.
3. On the client machine, delete the browser cache.
4. Configure the proxy settings to specify the VuGen machine as the proxy server.
Specify the machine address and port on which the LoadRunner proxy will listen.

Below are sample

Browser/OS	Path	Configuration
Internet Explorer	<ul style="list-style-type: none">• Internet Options > Connections > LAN Settings > Proxy serveror• Control Panel and IE Tools > Options menu	<ol style="list-style-type: none">a. Select Use a proxy server for your LANb. Specify Portc. Specify Address
FireFox	Tools > Options > Network > Advanced > Connection > Settings...	<ol style="list-style-type: none">a. Select Manual proxy configurationb. Specify HTTP Proxyc. Specify Portd. Check Use this proxy server for all protocols

5. Record the business process.
 - a. Navigate to your application.
 - b. Perform the steps of your business process you want to record.
6. Generate the script.

Select **Stop Recording** from either the **Recording Toolbar** on the client machine or the **Floating Recording Toolbar** on the VuGen machine. VuGen generates the script.

Note: It is common for business processes to use SSL communication even when not explicitly displaying a URL with HTTPS. In this case, a certificate may be required. Refer to Use Case 3 for more information on obtaining the certificate.

Use Case 2

You want to record a business process but you cannot install VuGen on the machine (or device) running the application. The client machine requires a proxy to access the Internet.

1. Create a new **Web - HTTP/HTML** script.
2. Start recording.
 - a. From the **Start Recording** dialog box, select **Recording Mode > Record > Remote application via LoadRunner Proxy**.
 - b. If necessary, change the port on which the LoadRunner proxy listens. The default is port 8888.
 - c. Select **Display recording toolbar on client machine**. This allows you to see and interact with the recording toolbar on the client machine.
 - d. Set the VuGen Internet Explorer proxy details:
In Internet Explorer, select **Tools > Internet Options > Connections**. Click **LAN settings** and enter the port and address of the client machine's Internet proxy.

Note: The **Use automatic configuration script** option is not supported.

- e. Select **Start Recording**.
3. On the client machine, delete the browser cache.
 4. Configure the browser settings to use the VuGen's machine IP and port.

Below are sample

Browser/OS	Path	Configuration
Internet Explorer	<ul style="list-style-type: none">• Internet Options > Connections > LAN Settings > Proxy server• Control Panel and IE Tools > Options menu	<ol style="list-style-type: none">a. Select Use a proxy server for your LANb. Specify Portc. Specify the VuGen IP in Address

Browser/OS	Path	Configuration
FireFox	Tools > Options > Network > Advanced > Connection > Settings...	<ul style="list-style-type: none"> a. Select Manual proxy configuration b. Specify HTTP Proxy c. Specify Port d. Check Use this proxy server for all protocols

5. Record the business process.
 - a. Navigate to your application.
 - b. Perform the steps of your business process you want to record.
6. Generate the script.

Select **Stop Recording** from the either the **Recording Toolbar** on the client machine or the **Floating Recording Toolbar** on the VuGen machine. VuGen generates the script .

Note: It is common for business processes to use SSL communication even when not explicitly displaying a URL with **https**. In this case, a certificate may be required. See **Use Case 3** for details on obtaining the certificate.

Use Case 3

Your application communicates using SSL.

1. Prepare to import the LoadRunner SSL certificate to the client machine.

Note: As an application developer, you can set certain policies on the server certificate when using SSL. However, only if the LoadRunner certificate conforms to the policy, can the client trust the server and the SSL connection be set.

2. Download the certificate, by navigating to `http://<computer name of VuGen machine>:port/proxyroot.cer` or `http://<ip address of VuGen machine>:port/proxyroot.cer`.

Note: If you experience security restrictions, navigate to `http://<computer name VuGen machine>:port/proxyroot.dat` or `http://<ip address of VuGen machine>:port/proxyroot.dat`. After downloading the certificate, change the `.dat` extension back to `.cer` to import the certificate.

3. Import the SSL certificate. The following table provides examples of the path for various browsers.
 - For Internet Explorer, select **Internet Options > Content > Trusted Root Certificate Authorities > Import**.

- For Firefox, select **Tools > Options > Advanced > Certificates tab > View Certificates > Authorities > Import**.

Use Case 4

You want to do a proxy recording of a local application that uses the system proxy, where VuGen and the client application are on the same machine.

1. Create a new **Web - HTTP/HTML** script.
2. Set the recording option.
Open the recording options (**Recording > Recording Options**) and select the **HTTP Properties > Advanced** node. Enable the **Use the LoadRunner Proxy to record a local application**.
3. Start recording.
 - a. Open the Start Recording dialog box.
 - b. In the **Recording mode** section, select **Record: Web Browser**.
4. Perform the business process.
 - a. Navigate to your application.
 - b. Perform the steps of your business process you want to record.

After recording

VuGen automatically resets the proxy back to its original setting after the recording. If the recording did not end in the normal way, for example, if your application crashed during recording, you may need to manually set the proxy back to its original value. To do so, go to **Internet Options > Connections > LAN Settings > Proxy server**.

If you are recording a Java application or a browser other than Internet Explorer, and the application is not using the system proxy settings, you will need to manually set the proxy of the application.

Import Actions to a Script

For Vuser types that support multiple actions, you can import actions into your script from another Vuser script.

You can import actions from Vusers of the same type only. Any parameters associated with the imported action will be merged with the script.

To import actions into the current script:

1. Select **Design > Action > Import Action**, or right-click the Solution Explorer and select **Import Action**. The Import Action into VuGen Script dialog box opens.
2. Click **Browse**. A list of the script's actions is displayed in the **Actions to Import** section.
3. Select the actions you want to include and click **Import**. The imported action(s) are displayed in the **Solution Explorer**.

Regenerate a Vuser Script

If you need to revert back to the originally recorded script, you can regenerate the script. This is ideal for debugging or fixing a corrupted script.

When you regenerate a script, VuGen removes all of the manually added enhancements to the recorded actions in the script. If you added parameters to your script, VuGen restores the original values. The parameter list, however, is not deleted; you can reinsert parameters that you created earlier. Note that regeneration only cleans up the recorded actions, but not those that were manually added.

Note: If your script was imported from a .zip archive file, make sure that it was archived with *all* files. If it was saved only with the runtime files, you will not be able to regenerate the script to its original recorded state. For details, see ["How to Work with .zip Files" on page 129](#).

To regenerate a Vuser script:

1. Initialize the regeneration.

Select **Record > Regenerate Script**. VuGen issues a warning indicating that all manual changes will be overwritten.

2. (Optional) Modify regeneration options .

Click **Options** to open the **Regenerate Options** dialog box.

- In a multiple protocol script, use the **General > Protocols** node to specify the protocols you want to record when the script is regenerated. For user interface details, see ["General > Protocol Recording Options" on page 166](#).
- To change the Script options, select the **General > Script** node and select or clear the appropriate check box. For user interface details, see ["General > Script Recording Options" on page 169](#).

Click **OK** to close the Regenerate Options dialog box.

3. Indicate whether to include imported actions.

If your script contains actions imported from another script, the dialog box will contain an option to delete imported actions during the regeneration.

If you chose to regenerate the script in a different language (using the **General > Script** options in the previous step), then non-recorded actions will automatically be deleted. Non-recorded actions include imported, renamed, or manually added actions.

Note: If a Flex or Java over HTTP Vuser script encounters errors during the code generation phase, VuGen shows the errors in the Error pane. The Error pane displays details about each error, as well as recommended actions. Follow the recommended actions and regenerate the script.


Start Recording Dialog Box

This dialog box enables you to record your business process.

To access	<ul style="list-style-type: none"> • Record > Record • [VuGen] Start Recording button
Important information	<p>This dialog box is dynamic and changes according to the options you select and the protocol you are using.</p> <ul style="list-style-type: none"> • To see all the options, click the More Options button in the top right of the dialog box. • To see only the basic options, click Fewer Options.
Relevant tasks	<ul style="list-style-type: none"> • "Record a Vuser Script" on page 135 • "Scripting Options Tab" on page 96 • "Record a Script via a Proxy" on page 216 • "Create a Vuser Script by Analyzing a Captured Traffic File" on page 685

User interface elements are described below:

All Protocols - recording (except Java)


UI Element	Description
Record into action	<p>The section of the script into which you want to record. You can choose one of the built-in action sections: vuser_init (for initialization steps), Action (for repeatable steps), or vuser_end (for sign off steps).</p> <p>You can also add your own action. Type the action name in the Record into action field and click the Add button . The new action is added to the script.</p>

UI Element	Description
Record	<p>The mode used to record your business process.</p> <ul style="list-style-type: none"> • Web Browser. For example, Web and Oracle NCA protocols record Web applications. <div> <p>Note: Recording intranet sites may not work with Microsoft Edge. For information on the workaround for this, see the troubleshooting entry for Recording on Microsoft Edge.</p> </div> <ul style="list-style-type: none"> • Windows Application. For example, the Windows Socket protocol records Windows applications. • Remote Application via LoadRunner Proxy. (For Internet protocols only) This option allows you to record traffic when VuGen cannot run on the client machine, such as Linux machines, Mac OS machines, and mobile devices. If you choose this mode, you can specify the following options: <ul style="list-style-type: none"> • LoadRunner proxy listens on port: The port on which the LoadRunner proxy will listen. • LoadRunner uses network interface. For Network Virtualization, enables you to select a network interface if there is more than one interface defined on the VuGen machine. • Display recording toolbar on client machine: Enables you to interact with the recording toolbar on the client machine. • Captured Traffic File Analysis. For details about creating a script using a captured traffic file, see Captured Traffic File Analysis below.
Application	<ul style="list-style-type: none"> • For Web Browser recording: Select one of the browsers detected on the machine. • For Windows Application recording: Specify the path of an executable file. <div> <p>Note: To run a batch file (a file with a .bat extension), specify <i>cmd.exe</i> with its path as the Application, and the batch file as the Program arguments.</p> </div>
URL address	<p>The starting URL address. This option is displayed only when you select the Web Browser recording mode.</p>

UI Element	Description
Program arguments	(Windows Application recording mode only) The command line arguments for the executable file specified in Recorded application . For example, if you specify plus32.exe (recorded application) with the command line options peter@neptune, it connects the user Peter to the server Neptune when starting plus32.exe.
Start Recording (For Internet protocols only)	You can record your business process either: <ul style="list-style-type: none"> • Immediately - Recording starts as soon as you click the Start Recording button. • In delayed mode - In the following instances, you may not want to record immediately: <ul style="list-style-type: none"> • You are recording multiple actions, in which case you only need to perform the startup in one action. • You want to navigate to a specific point in the application before starting to record. • You are recording into an existing script.
Working directory	For applications that require you to specify a working directory.
Recording Options	Opens the Recording Options dialog box. For user interface details, see "Recording Options" on page 141 .
Start Recording button	Begins to record your business processes based on the option selected above: Immediately or In delayed mode .

Captured Traffic File Analysis

UI Element	Description
Record into action	See All Protocols - recording above.
Record	Select Captured Traffic File Analysis . See All Protocols - recording above for details about the other options.
Captured file	Locate your pcap , saz (Fiddler), or har capture file. For details about creating a capture file in a Windows, Linux, or mobile environment, using an external tool such as Wireshark, see "Create a PCAP File" on page 831 .

UI Element	Description
Client side filter	<p>The IP address of the <i>client</i> whose traffic you want to examine. VuGen typically detects the client side filter by analyzing the capture file.</p> <div>  Tip: Use the recording options to set a server side filter (Recording > Recording Options > Network > Mapping and Filtering). </div>
SSL Attributes	<p>A list of the SSL attributes for the servers being analyzed.</p> <p>Use the Add, Edit, and Remove buttons to manage the entries.</p> <p>The Add button opens the Add SSL Attribute dialog box, allowing you to add a server and specify its IP address, port, certificate file, and password if required.</p> <p>This list is visible only after you select a pcap or saz capture file that contains SSL data and requires a certificate. This list is not available for har files—instead, configure the SSL through Fiddler.</p>
Start Recording button	Begins to analyze your captured file.

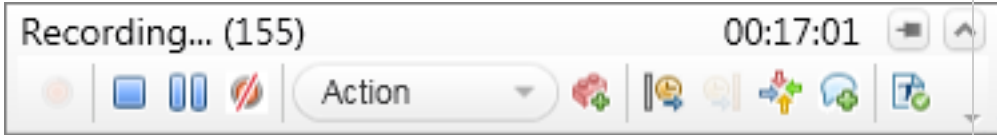



Java Protocols

UI Element	Description
Record into action	The section into which you want to record.
Record:	<ul style="list-style-type: none"> • Java applet to record a Java applet through Sun's applet viewer. • Java application to record a Java application. • Internet Explorer to record an applet within a browser. • Executable/Batch to record an applet or application that is launched from within a batch file or the name of an executable file. • Listener to instruct VuGen to wait for the batch file that initializes the configuration and runs an application before recording. This mode requires you to define the system variable <code>_JAVA_OPTIONS</code> as --Xrunjdkhook using <code>jdk1.2.x</code> and higher. (For JDK 1..x, define the environment variable <code>_classload_hook=JDKhook</code>. For JDK 1.6 set <code>_JAVA_OPTIONS</code> as -agentlib:jdhook.)
URL address	The URL to start recording (for Internet Explorer recordings)
Parameters	Any additional parameters that your application requires.





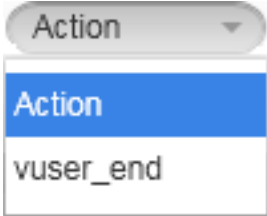






UI Element	Description
Record into action	The section into which you want to record.
Main Class	<p>The <i>complete</i> path of the Java class with the main method.</p> <p>Note: This option is only present for Java Application type applications.</p>
Applet path	This option is only present for Java applets.
Internet Explorer path	This option is only present for Internet Explorer type applications.
Executable\Batch	This option is only present for Executable\Batch type applications.
Working directory	A working directory is necessary only if your application must know the location of the working directory (for example, reading property files or writing log files). The default is the local LoadRunner/VuGen bin directory.
Recording Options	Opens the Recording Options dialog box. For user interface details, see "Recording Options" on page 141 .






Floating Recording Toolbar

The floating recording toolbar enables you to control the recording of Vuser scripts, and provides easy access to common script commands.

UI example	
To access	The floating recording toolbar appears when script recording begins.
Important information	<ul style="list-style-type: none"> The floating recording toolbar is dockable. For details, see "VuGen User Interface" on page 37. You can pin the toolbar with the  button. You can expand or collapse the toolbar with the  and  buttons.
Relevant tasks	"Creating or Opening Vuser Scripts" on page 113

User interface elements are described below:

UI Element	Description
	Continue recording the script after recording has been paused.
	Stop recording the script.
	Pause recording.
	Cancel the recording.
	Select an action to record into.
	Create a new action to record into.
	Insert a Start Transaction step into your script. For details, see "Transaction Overview" on page 323 .
	Insert an End Transaction step into your script. For details, see "Transaction Overview" on page 323 .
	Insert a Rendezvous point step into your script. For details, see "Rendezvous Points" on page 328 .
	Insert a comment into your script.
	Insert a Text Check step into your script (not available for all protocols). For details, see "Text and Image Verification (Web Vuser Scripts) - Overview" on page 694 .

UI Element	Description
	Displays: <ul style="list-style-type: none"> How many events have been recorded into your script. The time elapsed since recording began, excluding time the script was paused.
	Pin or unpin the recording toolbar.
	Display or hide the toolbar buttons.
	Hide the recording toolbar. The toolbar reappears when you refresh or navigate to the next page. Hiding the toolbar may be useful if the toolbar covers controls in the application being operated, thereby preventing access to the controls. <div> <p>Note: The Hide Toolbar button  appears for proxy recording only.</p> </div>

Files Generated During Recording

Assuming that the recorded script has been given the name **vuser** and is stored under **c:\tmp**, the following is a list of the more important files that are generated after recording:

File Name	Details
-----------	---------

vuser.usr	<p>Contains information about the Vuser type, AUT, action files, and so on.</p> <p>Example:</p> <pre>[General] Type=Oracle_NCA DefaultCfg=default.cfg BuildTarget= ParamRightBrace=> ParamLeftBrace=< NewFunctionHeader=0 MajorVersion=5 MinorVersion=0 ParameterFile=nca_test3.prm GlobalParameterFile= [Transactions] Connect= [Actions] vuser_init=init.c Actions=run.c vuser_end=end.c</pre>
vuser.bak	A copy of Vuser.usr before the last save operation.
default.cfg	<p>Contains a listing of all runtime settings as defined in the VuGen application (think time, iterations, log, web).</p> <p>Example:</p> <pre>[General] XlBridgeTimeout=120 [ThinkTime] Options=NOTHINK Factor=1 LimitFlag=0 Limit=1 [Iterations] NumOfIterations=1 IterationPace=IterationASAP StartEvery=60 RandomMin=60 RandomMax=90 [Log] LogOptions=LogBrief MsgClassData=0 MsgClassParameters=0 MsgClassFull=0</pre>

vuser.asc	The original recorded API calls.
vuser.grd	Contains the column headers for grids in database scripts.
default.usp	Contains the script's run logic, including how the actions sections run.
init.c	Exact copy of the Vuser_init function as seen in the VuGen main window.
run.c	Exact copy of the Action function as seen in the VuGen main window.
end.c	Exact copy of the Vuser_end function as seen in the VuGen main window.
vdf.h	A header file of C variable definitions used in the script.
\Data	The Data folder stores all of the recorded data used primarily as a backup. Once the data is in this folder, it is not touched or used. For example, Vuser.c is a copy of run.c.

Troubleshooting and Limitations for Recording with VuGen

Proxy recording

- If a client-side certificate is required during remote proxy recording, the dialog box requesting the certificate, opens on the VuGen machine, and not on the client machine.
- **Issue:** When recording a session in Chrome, the browser may appear to hang as it continually searches for external links.
Workaround: Manually set the environment's proxy settings in Chrome—do not enable **Automatically detect settings**.

Security Levels

Issue: "Trusted sites" appears in every recorded snapshot.

Solution: Open Internet Explorer at least once before recording a script in VuGen.

Troubleshooting missing steps

Issue: Your script is missing steps you recorded.

You encounter the following warning in the **Output Pane > Code generation** tab:

Warning: One or more responses are missing or have missing packets. Therefore, a step may appear to be missing in the script. This issue can be caused if the recording was stopped before all the responses were received. If the script is generated from a .pcap file, check if the file has missing packets.

This can be caused when you click **Stop Recording** before all the traffic has been received.

Steps to Resolve: Record the script again. Make sure all pages and resources have been downloaded before clicking the **Stop Record** button.

Recording on Internet Explorer 10

Issue: When recording on Internet Explorer (IE) 10, the browser uses cached pages, and may not record all of the steps.

Steps to Resolve: Each time you begin recording, configure IE 10 to always refresh Web pages from the server. After you begin a recording session, in IE, click F12 to open the Developer Tools pane. In this pane, usually located at the bottom of the browser window, select **Cache > Always refresh from server**.

Recording on Microsoft Edge

Issue: VuGen supports Microsoft Edge (run on Windows 10) to record Web protocols; however, by default, loopback calls are disabled in Windows 10. As a result, Edge is unable to access intranet sites through a proxy server, and consequently, VuGen fails to record intranet applications using Edge.

When you select **Microsoft Edge** as the Web browser in the ["Start Recording Dialog Box" on page 221](#), VuGen will try to enable loopback calls:

- If you have administrator privileges on the machine, VuGen can enable loopback calls, and you can proceed with intranet recording. (The loopback calls will be disabled again at the end of recording.)
- If you do not have administrator privileges, a warning is displayed. If you do not want to record intranet sites, then you can ignore the warning and proceed. If you do try to record intranet sites, Edge will not be able to navigate the sites.

Steps to Resolve: There are two workarounds that will allow recording of intranet sites; however, both options require administrator privileges and therefore pose security issues:

- Run VuGen as administrator each time an Edge recording is performed. When you select **Microsoft Edge** as the Web browser, VuGen will enable the loopback calls.
- Log in as administrator and enable the loopback call from the command line. Use the following command line setting:

checknetisolation loopbackexempt -a -p=S-1-15-2-3624051433-2125758914-1423191267-1740899205-1073925389-3782572162-737981194

Certificate warning message

When you open VuGen as a non-administrator user, during the recording process you may see a certificate pop-up warning message. The message is automatically closed and does not affect the recording.

Multi-Protocol recording

If you record a script in the **Init** section and then re-record in the **Actions** sections, compilation may fail. This happens because VuGen creates new header files with each code generation, removing the old ones.

Workaround: Re-record the new session in the same section as the first recording.

Overwriting of data

When recording a Web HTTP/HTML script using WebSockets, if you stop the recording and then resume the recording session, the new data overwrites the original data in the buffer. This is true even if you perform the second recording into a new action.

Firefox as default browser

If Firefox is set as the default browser, the **Use the default HTTP proxy settings** option (Runtime Settings > Internet Protocol > Proxy) does not work, and a direct connection is used.

FTP and Active SSL

FTP Active SSL mode is not supported for record or replay.

HSTS Web Recording

If you try to record an HSTS (HTTP Strict Transport Security) enabled site, using an SSL level other than 2/3, you will be unable to navigate within the site.

Workaround: Set the SSL level to SSL 2/3 (**Record > Recording Options > Network > Port Mapping > New Entry / Edit Entry**).

FTP Recording

An FTP recording may generate an empty script.

Workarounds: Perform one of the following:

- Configure the FTP server to include the string "FTP" in the welcome message.
- Open the **Network > Mapping and Filtering** recording option node. In the Port mapping area (upper section), click **New Entry**. In the Server Entry - Port Mapping dialog box, set the **Service ID** to **FTP** and specify the FTP server's port number.

64-bit Recording

In general, 64-bit applications ported from a 32-bit client version should work identically to the 32-bit client. There is a small risk that new clients will use the power of native 64-bit applications. For example, when using 64-bit long types for Identifiers in DB tables, the identifier value will be cut and the query will fail.

The following guidelines apply:

- The environment for 64-bit recording must be a Windows 7 x64 or Windows 8 x64 (Windows 8 x64 added in Service Pack 11.52), and a 64-bit Application Under Test (AUT).
- Recording on 64-bit operating system for 32 and 64-bit applications (running as a 64-bit application) is supported.
- You cannot record a page requiring a client certificate with 64-bit version of Internet Explorer.
- For the Java Over HTTP protocol: JVM 32-bit is required for replay.
- Oracle 2-Tier: Both 32-bit and 64-bit clients need to be installed (the 32-bit client is required for

replay).

- For the .NET protocol: There are two available 64-bit types for .NET applications (AnyCPU and pure 64-bit). LoadRunner only supports AnyCPU. There is currently no solution for pure 64-bit applications.

For replay, LoadRunner uses the same AnyCPU dlls that were used for Recording.

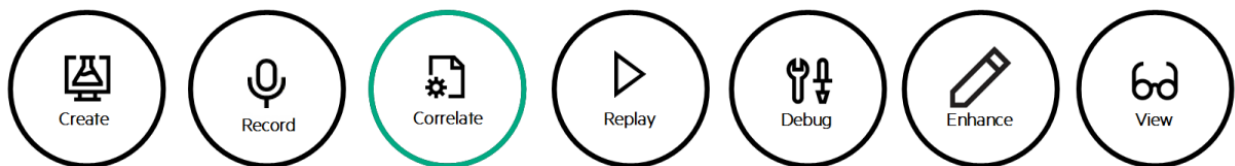
Note: With LoadRunner 11.50 and later, .NET Framework 4 or later should be installed.

Correlating

Correlation Overview

Creating a Vuser script includes the steps shown below. This topic provides an overview of the third step, correlating a Vuser script.

Select an image to learn more.



Web pages sent from the server to the browser often contain dynamic data that the client must return in later requests. The process of locating, extracting and replacing recorded dynamic values with values valid at replay is called "correlation".

Common examples of dynamic data are:

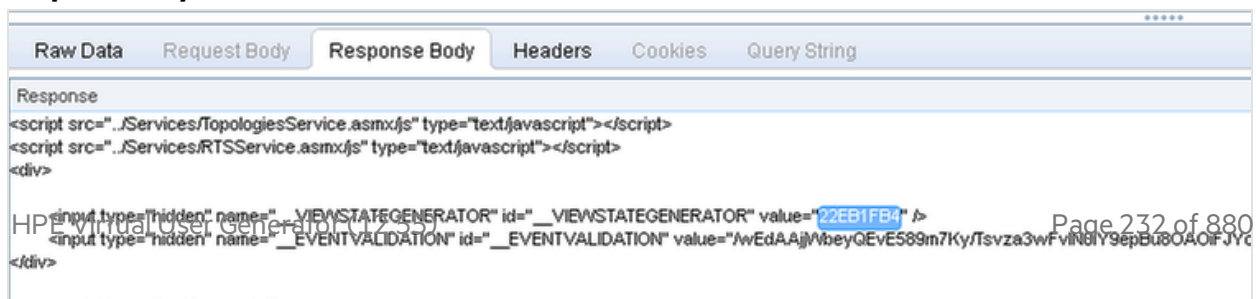
- Session ID
- Timestamp
- Customer ID
- Authentication token

This information generally becomes invalid when the session is completed. At the next session, the same items are sent, but the values are different.

After recording the business process in LoadRunner, the script might contain dynamic values in the arguments of the generated API functions. When the script is replayed, these recorded values are sent to the server. However, the web server rejects them because they are not valid in the replay session. Generally, the test fails when this happens.

Example of a dynamic value as received from the server and as it appears in the recorded script:

Response Body



Resulting script

```

19
20     web_add_auto_header("X-MicrosoftAjax",
21         "Delta=true");
22
23     web_add_auto_header("X-Requested-With",
24         "XMLHttpRequest");
25
26     web_add_auto_header("X-XSRF-Header",
27         "");
28
29     web_submit_data("Login.aspx_2",
30         "Action=http://mypcserver/LoadTest/General/Login.aspx",
31         "Method=POST",
32         "RecContentType=text/plain",
33         "Referer=http://mypcserver/LoadTest/General/Login.aspx",
34         "Mode=HTML",
35         ITEMDATA,
36         "Name=__VIEWSTATE", "Value="
37         "/wEPDwUKMHTc3NTA3OTE1Nk8kZBYCZg9kFgQCAQ9kFgICAQ8WAh4EVGV4dAUyPG1ldGEgaHR0cC1lcXVpdj0iWC1VQS1Db21wYXRpYmxlIiI
38         "V2lkdgGcHglDb2x5XzZWRoHglNYXhIZWlnaHQkE4e8khlaWdobSAAAAAABCLQAEAAAAeCE1pbldpZHRoAhQeDk9yaWdpbmFsSGVpZ2
39         "Rpc2FibGVkQ2hlY2tCb3hXcmFwcGVyHxMCAmQWBAIDdw8Bh4TQXNzb2NpYXRlZENvbnRyb2xJRGUfEgUdbGJsRm9yQ2hrQm94IERpc2Fi
40         "RkFVTfQe8VZhbHVlBQdERUZBVUxUHghTZWx1Y3RlZGdkZA8UKwEBZhY8B8dUZWxlcm1rLldlYi5VSS5SYWRDb21ib0JveE10ZW0sIFRlbG
41         "kuUmFkQ29tYm9Cb3hjdGVtLCBUZWxlcm1rLldlYi5VSS5wVmVyc2lwbj0yHDE0LjEuNDZAZLjQwLDB0dWx0dXJlPW5ldXRyYWwsIFB1Ym9p
42         "cgUby3RsMDAKUGFnZUNvbnRlbnQkdXBwZXJQYW51BSRjdGwwMCRQYwdlQ29udGVudCRjYkF1dG9Nb2dpbiRjaGtCb3gFHGN0bDAnJF8hZ2
43         "Name=__VIEWSTATEGENERATOR", "Value=22EB1FB4", ENDITEM,
44         "Name=__EVENTVALIDATION", "Value=/wEdAAh6Ar/0pHDJD2/0DpEmoIcua3wFv1N0iY9epBu80A0iFJYd12hp8axRYEbe5n4K%2BTf
45         "Name=__ASYNCPOST", "Value=true", ENDITEM,
46         "Name=ctl00$PageContent$btnLogin", "Value=Login", ENDITEM,
47         LAST);
48
49     lr_think_time(6);
50

```

To enable successful replay, the script must:

- Locate the dynamic data in the server responses, including headers.
- Extract and save that dynamic data.
- Replace the hard-coded dynamic data from the recording session with the data from the current replay session.

Dynamic data is located using boundaries definitions, regular expressions, attributes, XML queries, or JSON queries as appropriate. The data is extracted and the value is saved to a parameter.

The parameter is used instead of the recorded value in later requests to the server.

Some of the correlation is handled automatically by LoadRunner. You can configure the automatic correlations. For more information, see ["Automatic Correlation" on page 236](#) and ["Automatic Correlation Configuration" on page 237](#).

Dynamic data that is not correlated automatically requires your intervention. For more information, see ["Manually Correlate Scripts" on page 242](#) and ["Design Studio" on page 241](#).

If the script still fails on correlation problems, you can use the advanced techniques. For more information, see ["Advanced Correlation Techniques" on page 251](#).

Note: Although parametrization and correlation both use LoadRunner parameters, the purpose is different.

- Parameterization is used to vary the requests based on a list of values you provide. Parameterization is not dependent on preceded responses from the server.
- Correlation uses data from the server in subsequent requests so that the server does not reject the request.

Correlation Tab [Design Studio] Overview

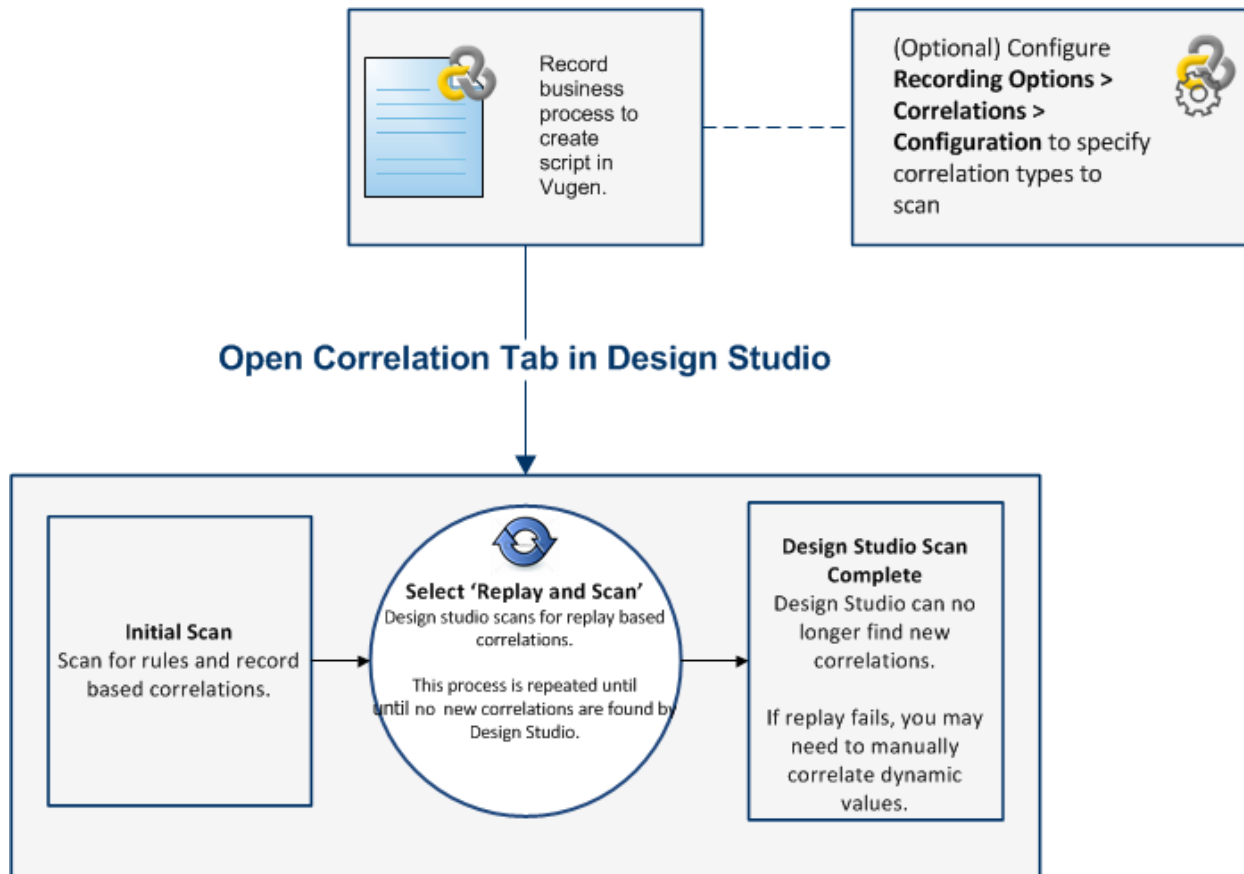
The Correlation tab enables you to correlate and manage dynamic values in your web-based Vuser scripts. To learn more about correlation concepts, see ["Correlation Overview" on page 232](#).

With the **Correlation tab** you can:

- Scan for correlations using rules, record based, and replay based engines
- Correlate both raw and formatted data
- Add and edit rules
- Undo correlations
- Review details pertaining to a specific dynamic value in a snapshot

When you record a script using a web-based protocol, many of the values change dynamically each time a request is sent to the server. An example of a dynamic value is a `sessionID` which may include a date and time stamp of when the web session was opened. To learn more, see ["Design Studio \[Correlation Tab\] Dialog Box" on page 242](#)

The following flow chart illustrates the process for correlating values in your script using the Correlation tab:



As you can see from the flow chart, the Correlation tab scans for dynamic values using different processes.

Correlation Types

Design Studio uses three processes to automatically find dynamic values that may need to be correlated.

- Rules
Design Studio first scans for dynamic values that are defined by rules, if the rules scan has been enabled. To learn more, see ["Correlation Rules" on the next page](#).
- Record
Design Studio scans for dynamic values after code generation. This method can find a significant percentage of dynamic values in your script.
- Replay
Design Studio scans for dynamic values after replay. This method may need to be repeated several times.

You can select which scan types the Correlation tab should use by configuring **Recording Options > Correlations > Configuration**. In general, it is recommended to enable all scan types.

The following table explains the expected behavior at various script states:

Script State when opening the Correlation tab	Behavior in the Correlation tab (All scan types enabled)
Script contains recorded data.	<p>When Design Studio is opened, it will scan for rule and record based correlations.</p> <p>You can then replay and scan for replay based correlations. Repeat this process until the Design Studio no longer finds new correlations.</p>
Script contains recorded data and has been replayed.	<p>When the Correlation tab is opened, it will scan for all correlation types.</p> <p>You can then replay and scan for additional replay based correlations. Repeat this process until Design Studio no longer finds new correlations.</p>

Correlation Rules

If you know the dynamic values that need to be correlated before recording, you can create correlation rules that will automatically identify those values while you record. If "**Automatically apply correlation rules**" is selected in the **Recording Options > Correlations > Configuration** node, values found based on rules will automatically be correlated. Additionally, there are some correlation rules that come pre-defined in VuGen for supported application servers. You can enable or disable rules in "[Correlations > Rules Recording Options](#)" on page 151.

Snapshot Details and Occurrences

Design Studio provides details on each snapshot step that contains dynamic values. These details can help you determine which values to correlate in your script. In addition to the snapshot details, the Correlation tab, displays all occurrences of the dynamic value in your script. You can select specific occurrences to correlate or correlate all. For details, see "[Design Studio \[Correlation Tab\] Dialog Box](#)" on page 242.

Automatic Correlation

LoadRunner can detect dynamic values in the script and suggest their replacement with parameters. For automatic correlation, LoadRunner uses rule-based, recording-based, and replay-based correlations.

Rule-based correlation

LoadRunner comes with a set of predefined, extendable rules for how to correlate dynamic values in well-known environments.



Example: An example of a rule in pseudo-code might be:

Find text in the response headers between "Timeout: " and the next white space and save it in parameter "Timeout".

During code generation, LoadRunner scans server responses for values that match the Correlation Rules. When a matching value is found, LoadRunner adds a step to extract the value and save it in a parameter.

Then, the script code is scanned for the value and matches are replaced with a reference to the parameter.

For information about tuning the rules-based correlation, see ["Automatic Correlation Configuration" below](#).

Recording-based correlations

After code generation, the script is scanned for additional correlation candidates in the client request steps. If a candidate was received from the server before it is sent by the client, it is suggested as a correlation. The suggestion includes the exact description of where and how to extract the dynamic value from the server response and where to replace data with a parameter.

You can accept suggestions in the ["Design Studio" on page 241](#).

Replay-based correlations

Replay the script at least once. It is not a problem at this stage if some steps fail.

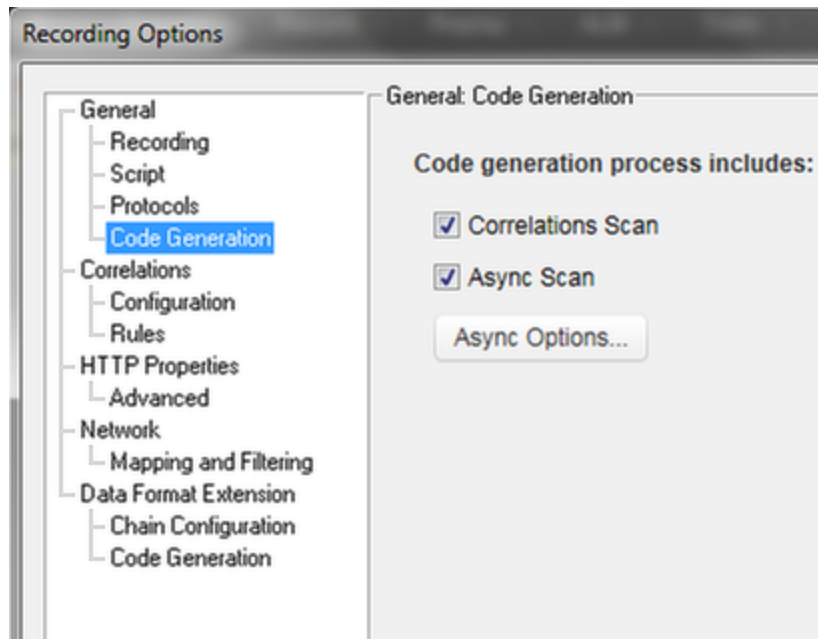
The Replay-based scan compares the server responses before the failures for mismatched values. If such mismatched values are used in the script, they are suggested as correlations.

Repeat this process until the last failure is detected and correlated.

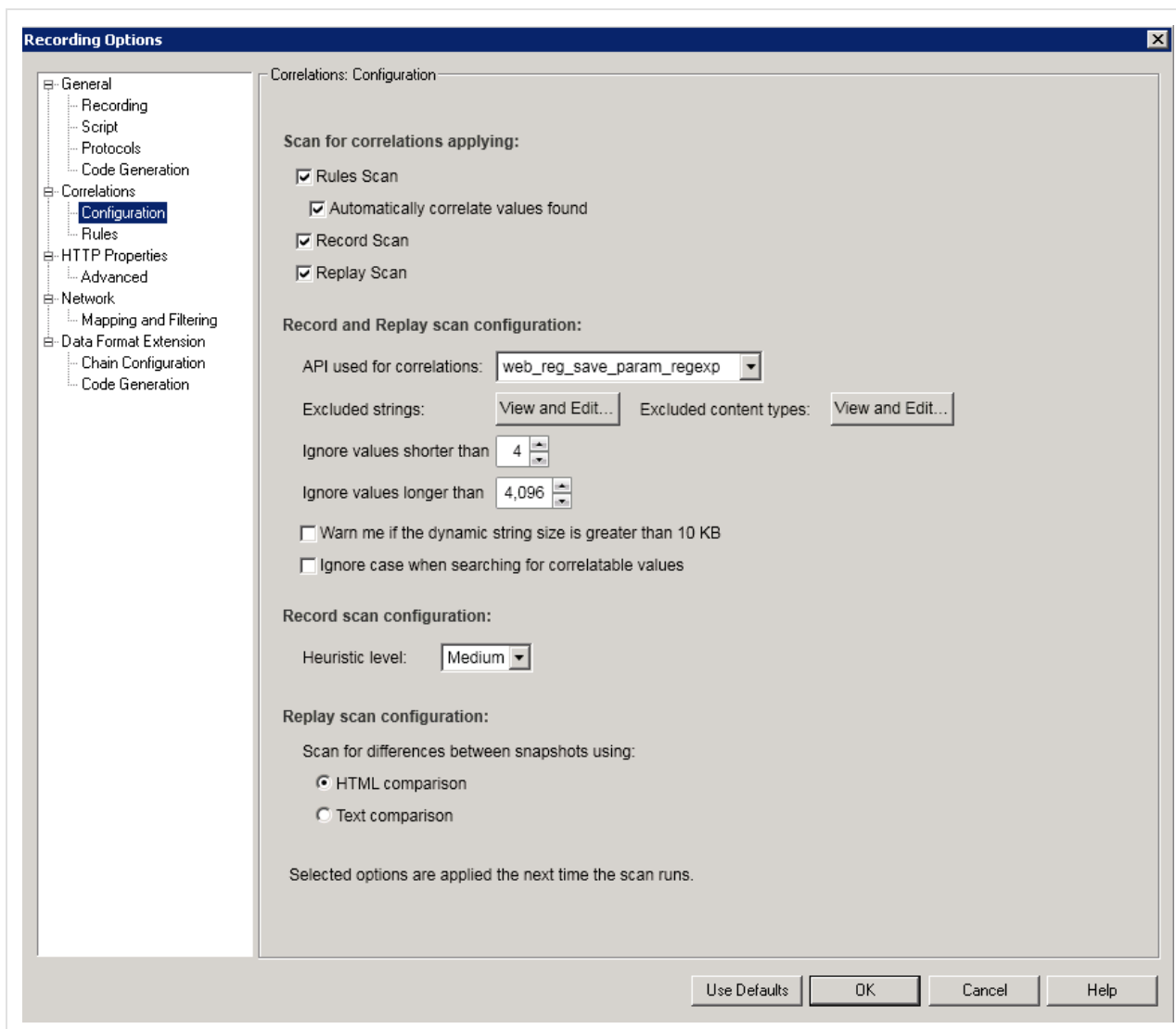
Automatic Correlation Configuration

By default, all correlation methods are enabled for all LoadRunner users. It is possible to change the correlation settings in the Recording Options dialog.

Automatic correlations are enabled by checking the **Correlation Scan** checkbox under the **General > Code Generation** node, as shown in the following screenshot:

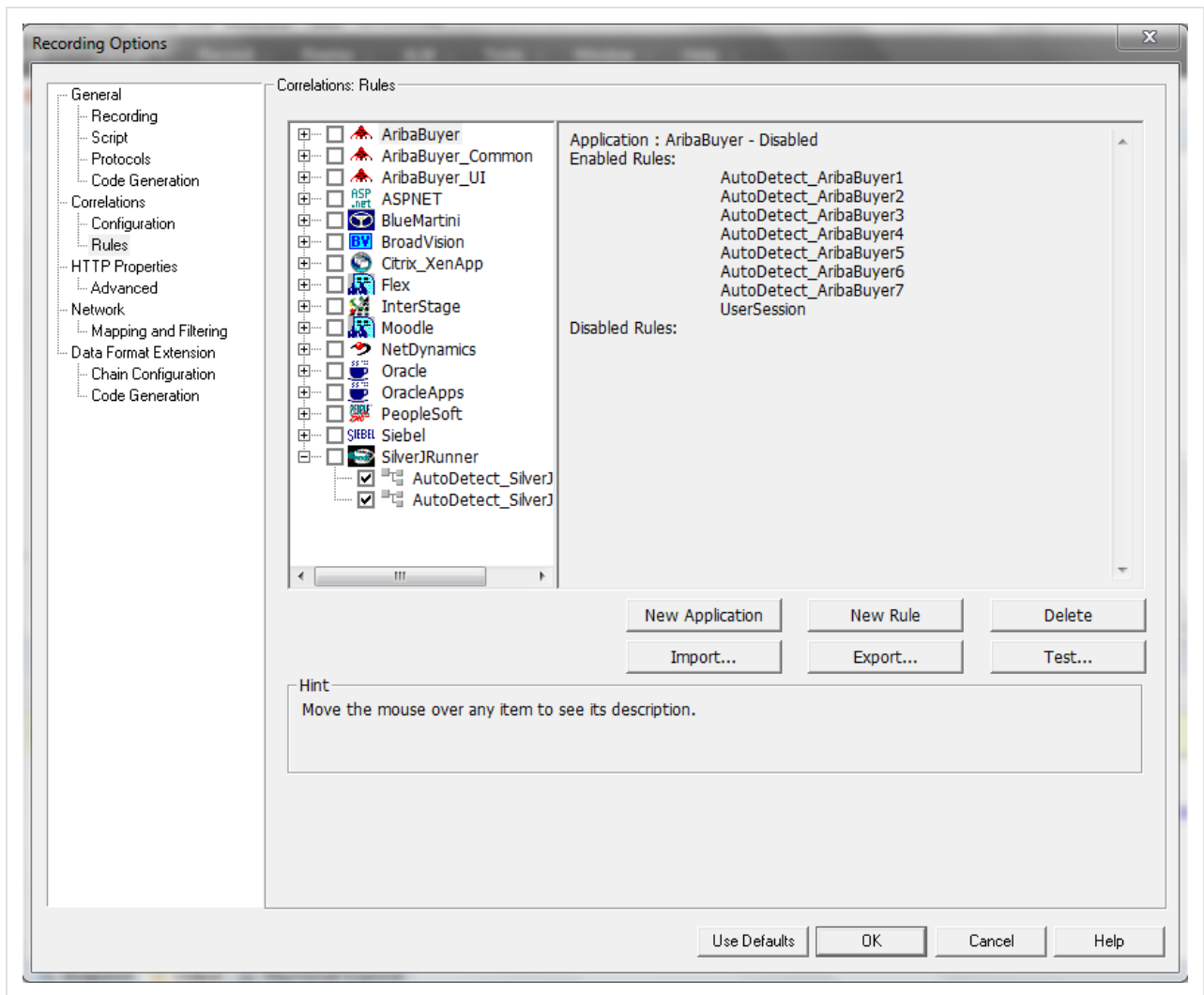


To define preferred settings, go to the **Correlations --> Configuration** node, as shown here:



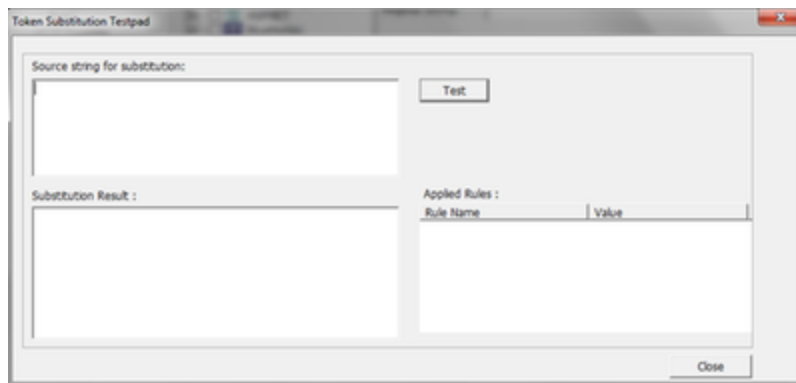
Correlation rules can be found under the **Correlations --> Rules** node. The rules are organized in logical groups (“applications”) that represent testing environments like ASP.NET or Citrix.

Correlation rules define where to search and where to replace the dynamic value, extraction method (boundary, regular expression or other) and provide meaningful parameter names to be used in the script for correlations, as shown in the next screenshot:



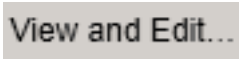

Correlation rules can be modified and expanded with new rules. You can test correlation rules before applying them with the Token Substitution Testpad.

You can also exchange and share rules using the Export/Import functionality:

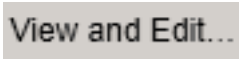




Exclude Strings or Content Types from the Correlation Scan

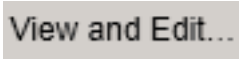


By default, the correlation engine scans all plain and html text searching for correlations. Some of the candidates found may not be real correlations. To enhance correlation accuracy, you can configure VuGen to ignore certain text strings, regular expressions, or content types.

1. To exclude a specific text string
 - a. Select **Record > Recording Options > Correlations > Configuration**.
 - b. Click the  button adjacent to **Excluded strings**.
 - c. Click the  button to open the **Add string to exclude** dialog box.

For example, enter `getCachedId` to exclude the string `getCachedID` as a correlation candidate.

2. To exclude matches of a regular expression
 - a. Select **Record > Recording Options > Correlations > Configuration**.
Click the  button adjacent to **Excluded strings**.
 - b. Click the  button to open the **Add string to exclude** dialog box.
 - c. Enter a regular expression and check the **Regular Expression** box or select the  button to view and select from a list box of regular expression character classes and complete the regular expression.

For example, enter `^navurl:.*` to exclude strings such as `navurl:\\any_char`, `navurl:1234` as correlation candidates..

3. To exclude content types
 - a. Select **Record > Recording Options > Correlations > Configuration**.
Click the  button next to **Excluded content types** to open the **Excluded Content Type List** dialog box. The list shows the content types that are automatically excluded.
 - b. Click the  button to add a new entry.
4. Delete an excluded item
 - a. Highlight an item in the **Excluded String List** or **Excluded Content Type List** dialog boxes.
 - b. Click the  button.

Design Studio

The Design Studio serves as a single entry point for all correlation functionality.

The Design Studio is displayed automatically when code generation has finished. It can also be reached manually by clicking on the "Design Studio" button in the toolbar or is accessible via the Design menu.

The Design Studio enables you to manage dynamic values detected by all correlation scans in the unified form and provides complete information about the location, extraction method and the expected replacements in the script.

You can examine the details of the “correlation candidate” and improve the configurations prepared by correlation engines if necessary (for example, correlation boundaries or parameter name).

With Design Studio, you can correlate dynamic values in the script, create correlation rules, undo correlations or discard some correlation suggestion as irrelevant (add these value to excluded strings).

For more information about the Design Studio, see ["Design Studio \[Correlation Tab\] Dialog Box"](#) below

Manually Correlate Scripts

If the scan for correlation does not resolve all correlation-based errors in your script, you can attempt to manually correlate your script as follows:

1. Search for values that need correlation manually. There are a number of ways to manually search for values that need correlation. For details, see ["Search for Values that Need Correlation" on page 251](#).
2. Correlate the value by doing one of the following:

- Correlate from snapshots. Highlight the value to correlate, right-click, and select **Create Correlation**.

When a value is correlated, VuGen adds the correlation parameter and saves the original value in a comment in the script.

```
252 | /* Correlation comment - Do not change! Original value='1' Name ='CorrelationParameter' Type ='Manual' */
253 |   lrc_save_rs_param(_Recordset_45,
254 |     1,
255 |     2,
256 |     0,
257 |     "CorrelationParameter");
```

- Manually add correlation functions. Manually insert the relevant correlation functions into your script.




See also:

- ["Web Manual Correlations" on page 271](#)
- ["Web Manual Correlation in Code" on page 272](#)
- ["Winsock Manual Correlation" on page 276](#)

Design Studio [Correlation Tab] Dialog Box

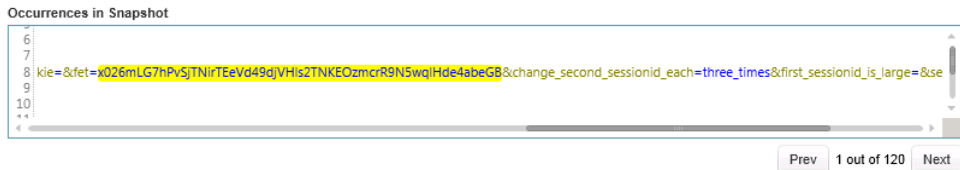
This dialog box enables you to scan for, correlate, and view information about dynamic values in your script.

To access	<p>Click the  Design Studio button on the VuGen toolbar.</p> <p>The button is enabled only when you have a recorded script in the Solution Explorer.</p>
Important information	"Correlation Tab [Design Studio] Overview" on page 234
Relevant tasks	"Correlate Scripts Using Design Studio" on page 250

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
Correlation Tab	
Replay and Scan	Design studio scans for dynamic values using all enabled types: rule, record, and replay.
Correlate	Replace a dynamic value in the script with a correlation parameter.
Add as Rule	<p>Add dynamic value definition as a rule.</p> <p>Rule name. Enables you to specify a rule name.</p> <p>Application Name. Enables you to associate the rule to a specific application.</p> <p>For details, see "Correlations > Rules Recording Options" on page 151.</p>
Undo Correlation	Replace the correlation parameter with the original dynamic value.
Discard	<p>Delete the selected dynamic values from the correlation grid. You can only use the discard action when the dynamic value has a status of New.</p> <p>In addition, this action adds the text to the list of excluded strings. You can edit the list in Recording Options > Correlation > Configuration > Excluded string list.</p>
View	<p>Enables you to filter values found for correlation by the following types:</p> <ul style="list-style-type: none"> • All • New • Correlated
Correlation Grid	
Displays details about each dynamic value in the script	

UI Element	Description
Type	<p>Displays which engine found the dynamic value for correlations:</p> <ul style="list-style-type: none"> • Record • Rules • Replay • Manual
Found/Replace	<p>Displays information about the number of dynamic values found with the same definitions. Since you can perform partial correlation, meaning you can replace specific occurrences, the information displayed depends on if you have correlated the value or not.</p> <ul style="list-style-type: none"> • Before you correlate Number of values that can be replaced/ Number of values found • After you correlate Number of values that have been replaced/Number of values found.
Status	<p>Displays correlation status of the dynamic value from the script:</p> <ul style="list-style-type: none"> • New • Correlated
Text in Response	Displays the string of the dynamic value from the script.
Correlation Parameter	Displays the correlation parameter name of the dynamic value.
Correlation Details Chevron Displays details about the dynamic value in the snapshot/script	
Original Snapshot Step Tab	
Step in Script Details	
Name	Displays the step name in the script where the dynamic value was found.
Line	Displays the line of the script where the dynamic value was found.
Action Name	Displays the name of action from the script where the dynamic value was found.
Description	Displays a description of the step.
Correlation Definition Details	

UI Element	Description
Type	<p>Display API function that will be used to correlate the value.</p> <p>Attribute based: web_reg_save_param_attrib</p> <p>Regular expression: web_reg_save_param_regexp</p> <p>Boundary based: web_reg_save_param_ex</p>
Definition	<p>Displays the definition of the dynamic value.</p> <ul style="list-style-type: none"> • Attribute Based. Dynamic value correlation is defined by an input text value. Returns a warning if the text is not found. • Regular Expression. Dynamic value correlation is defined by a regular expression. A regular expression is a special text string for describing a search pattern. • Boundary based. Dynamic value correlation is defined by left and right boundary text strings.
Apply Definition	<p>Enables you to select which definition to apply to the dynamic value. You can scroll through the definition of the dynamic value in Occurrences in Snapshot by clicking Prev or Next buttons.</p>
Occurrences in Snapshot	<p>Record snapshot. Displays all the occurrences of the dynamic value in the record snapshot once the script as been replayed. You can scroll to view each occurrence in the snapshot.</p>  <p>Replay snapshot. If the scan type of Replay has been selected, Design Studio displays all the occurrences of the dynamic value in the replay snapshot once the script as been replayed. You can scroll to view each occurrence in the snapshot.</p> <p>Note: Once the value has been correlated, the replay snapshot will be blank. If you modify the Correlation Definition, the replay snapshot will be blank.</p>
Correlation Occurrences Tab	
Occurrences in Script	<p>Displays the occurrences of the dynamic value in your script. You can correlate all the values or select individual values to correlate by selecting the check box adjacent to the occurrence.</p>


UI Element	Description
Options	Opens the Recording Options dialog box. For details, see: <ul style="list-style-type: none">• "Correlations > Configuration Recording Options" on page 149• "Correlations > Rules Recording Options" on page 151

Modify Correlation Definitions

You can modify correlation definitions to help eliminate dynamic values that do not require correlation. These tasks describe how to modify boundary based, regular expression, and XPath query correlation definitions for record or response correlation.

Note: These methods only apply to modifying correlations within your script. They do not affect the correlation rules.

Modifying Boundary Based Correlation Definitions

1. Click the  **Design Studio** button on the VuGen toolbar.
2. Select a dynamic value from the correlation grid and expand **Details**.
3. Edit the **Left Boundary** or **Right Boundary** under the **Correlation Definition** section. You can modify the definition by adding or deleting text.

Correlation Definition

Type: **Boundary Based**

Left Boundary:

Right Boundary:


Apply this Definition

4. Click **Apply this Definition**.

The **Apply this Definition** button will not be enabled unless the modified boundary definition occurs in the snapshot and the script.

Note: If you do not apply the definition before selecting another dynamic value in the grid, your changes will be lost. If you select **Replay & Scan** before correlating your value with the modified definition, your changes will be lost.


Modifying Regular Expression Correlation Definitions

1. Click the  **Design Studio** button on the VuGen toolbar.
2. Select a dynamic value from the correlation grid and expand **Details**.
3. Edit the **Regular Expression** under the **Correlation Definition** section.
4. Click **Apply this Definition**.

The **Apply this Definition** button will not be enabled unless the modified boundary definition occurs in the snapshot and the script.

Note: If you do not apply the definition before selecting another dynamic value in the grid, your changes will be lost. If you select **Replay & Scan** before correlating your value with the modified definition, your changes will be lost.


Modifying XPath Correlation Definitions

1. Click the  **Design Studio** button on the VuGen toolbar.
2. Select a dynamic value from the correlation grid and expand **Details**.
3. Edit the XPath definition under the **Correlation Definition** section.
4. Click **Apply this Definition**.

The **Apply this Definition** button will not be enabled unless the modified boundary definition occurs in the snapshot and the script.

Note: If you do not apply the definition before selecting another dynamic value in the grid, your changes will be lost. If you select **Replay & Scan** before correlating your value with the modified definition, your changes will be lost.

Modifying Winsocket Correlation Definitions

1. Winsocket dynamic values are correlated from the snapshot. To access, select the relevant step in the **Step Navigator**, and view the step in the **Snapshot pane**. The Winsocket protocol has both a hex and text snapshot.
Right click the value in the snapshot and select **Create correlation** or **Create boundary correlation**. This will open the  **Design Studio** window.
2. Select a dynamic value from the correlation grid and expand **Details**.
3. If you selected **Create correlation**, edit the Data Range in the **Correlation Definition** section. If you selected **Create boundary correlation**, edit the left of right boundary.

Correlation Definition

Type: Data Range

Offset: 82

Length: 4

Apply this Definition

Correlation Definition

Type: Boundary Based

Left Boundary: NAME="f

Right Boundary: \" value

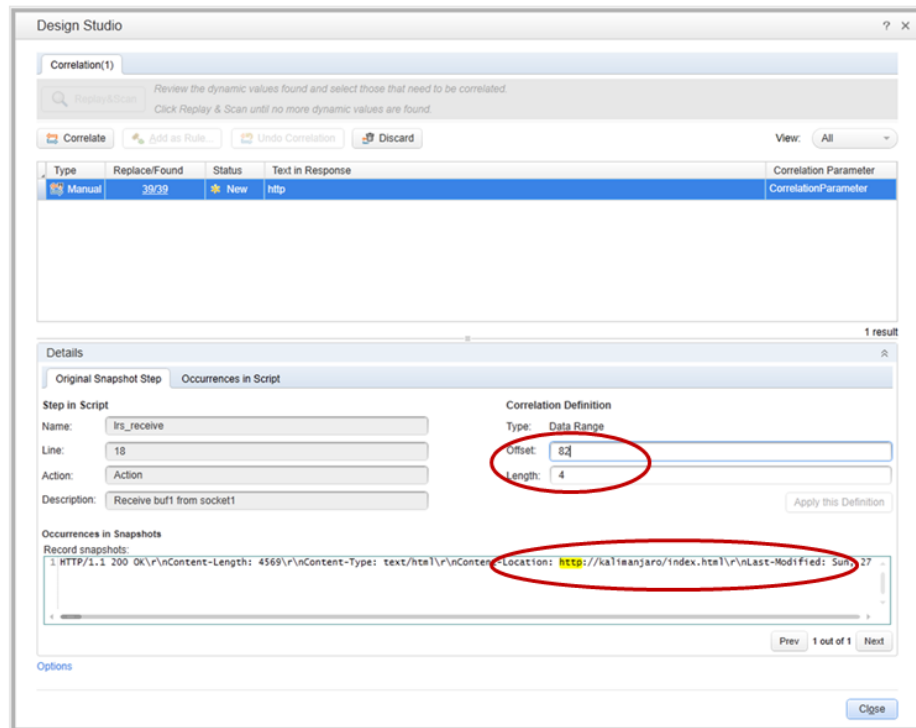
Apply this Definition

4. Click **Apply this Definition**.

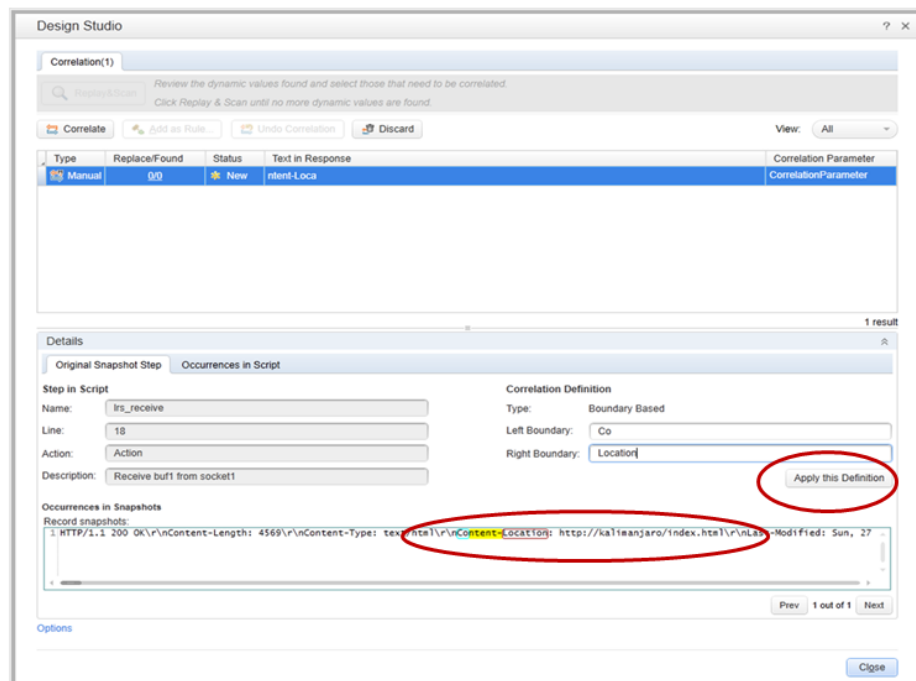
The **Apply this Definition** button will not be enabled unless the modified boundary definition occurs in the snapshot.

View the following images that display both a Data Range definition and a Boundary definition.

Data Range definition correctly modified



Boundary base definition correctly modified



Note: If you do not apply the definition before selecting another dynamic value in the grid, your changes will be lost. If you select **Replay & Scan** before correlating your value with the modified definition, your changes will be lost.

Correlate Scripts Using Design Studio

1. Prerequisites:

a. Record a script using one of the following protocols:

- Web HTTP/HTML
- Flex
- RTMP/RTMPT
- Citrix
- SAP - Web
- Oracle NCA

Note: You can correlate the Web HTTP/HTML steps only within your Oracle NCA script when the Web HTTP/HTML protocol is active. To activate, select **Recording Options > Protocol > Active Protocols > Web HTTP/HTML**.

For additional information, see [How to Correlate Scripts - Oracle NCA](#)

b. Verify that **Record > Recording Options > Correlations > Configuration** has all scan types enabled.

Perform correlation scan using:

- ☒ Rules Scan
 - ☐ Automatically apply found correlations
- ☒ Record Scan
- ☒ Replay Scan

2. In the Correlation tab, click the **Design Studio** button to perform an initial scan. This will open the Design Studio dialog box which will scan for response (or record based) correlations and apply correlation rules. The progress bar in the dialog box indicates if the initial scan was successful. For details on the Correlation tab, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

Note: If no dynamic values for correlation are found, a warning is displayed, advising you to check your recording options. Verify that **Record > Recording Options > Correlations >**

Configuration has all scan types enabled.

3. Select which values to correlate by highlighting the value in the grid and clicking the **Correlate** button.

When a value is correlated, VuGen adds a **web_reg_save_param_*** function, and saves the original value in a comment in the script.

You can examine the details of the correlation by expanding the Details Chevron in the dialog box. For details, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

4. Click the **Add as Rule** button to add a rule type . In addition, you can click the **Edit rule** button to view and edit the corresponding rule if the dynamic value was correlated by a rule. For details, see ["Correlations > Rules Recording Options" on page 151](#).
5. Click the **Replay and Scan** button. The Correlation tab progress bar indicates if additional correlations have been found. Again, you can select which values you would like to correlate by highlighting the value in the grid and clicking the **Correlate** button. You may need to repeat this step several times.
6. When Design Studio no longer finds new correlations, the progress bar will display **Design Studio Scan Complete**.



Tip: If Design Studio does not resolve all correlation-based errors, try to resolve them using manual correlation. For details, see ["Manually Correlate Scripts" on page 242](#).

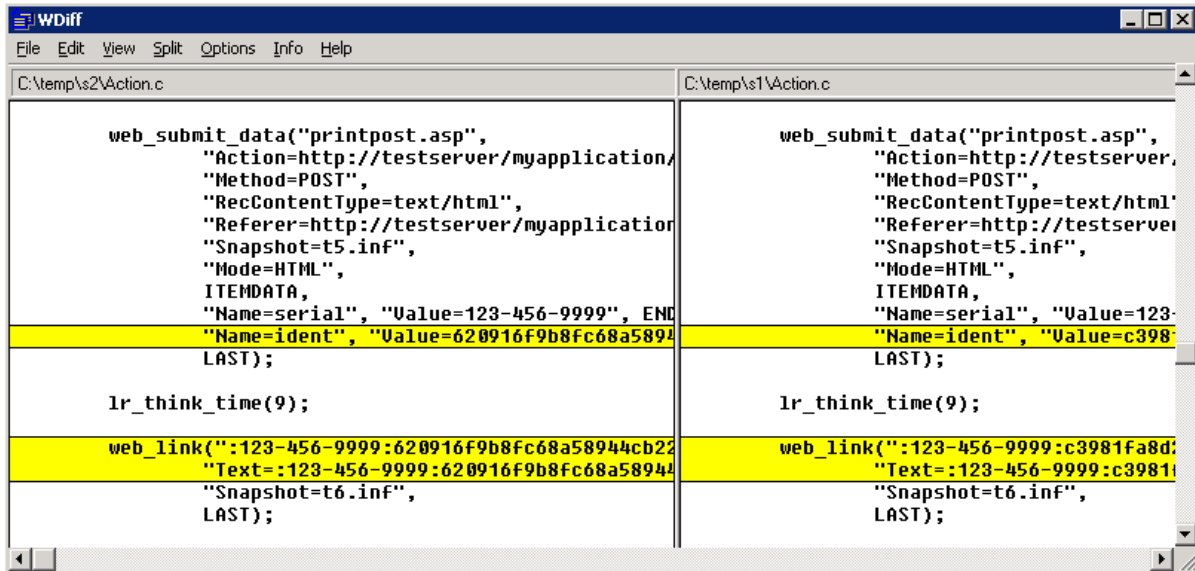
Advanced Correlation Techniques

If your script continues to fail on correlation problems after you have finished the tasks describe above, you may need to use the advanced techniques described in this section.

Search for Values that Need Correlation

Search by Comparing Scripts

1. Record a script and save it.
2. Create a new script and record the identical operations. Save the script.
3. Select **Tools > Compare with Vuser** to compare the scripts. For more details, see ["Compare Scripts Side-by-Side" on page 115](#).
4. Differences in the script are highlighted. Review the differences to determine which ones may require correlation.



Note: *WDiff* is the default utility, but you can specify a custom comparison tool. For more information, see ["Compare Scripts Side-by-Side" on page 115](#).

Replay Log Search

1. Scan the script in script view for strings that may need correlation such as hash strings, random strings, session ID's, and so on.
2. Search the generation log for the first time that the string appears (this is the response from the server).
3. Search the extended replay log for the same response. Check to see if this response contains a different string within the same boundaries as the original suspected string. If yes, this string requires correlation.

Wdiff Correlation Utility

The Wdiff Utility lets you compare recorded Vuser scripts and replay results to determine which values need to be correlated.

Note: *WDiff* is the default utility, but you can specify a custom comparison tool. For more information, see ["Compare Scripts Side-by-Side" on page 115](#).

To use *WDiff* effectively, you record the identical operation twice, and compare the scripts (or data files for WinSock). *WDiff* displays differences in yellow. Note that not all differences indicate a value to correlate. For example, certain receive buffers that indicate the time of execution do not require correlation.

Modifying Saved Parameters

After you save a value to a parameter, you may need to modify it before using it in your script. If you need to perform arithmetical operations on a parameter, you must change it from a string to an integer using the **atoi** or **atol** C functions. After you modify the value as an integer, you must convert it back to a string to use the new variable in your script.

In the following WinSock example, the data at offset 67 was saved to the parameter, **param1**. Using **atol**, VuGen converted the string to a long integer. After increasing the value of **param1** by one, VuGen converted it back to a string using **sprintf** and saved it as a new string, **new_param1**. The value of the parameter is displayed using **lr_output_message**. This new value may be used at a later point in the script.

```
lrs_receive("socket2", "buf47", LrsLastArg);lrs_save_param("socket2",  
    NULL, "param1", 67, 5);  
lr_output_message ("param1: %s", lr_eval_string("<param1>"));  
sprintf(new_param1, "value=%ld", atol(lr_eval_string("<param1>")) + 1);  
lr_output_message("ID Number:%s" lr_eval_string("new_param1"));
```

Correlation Specifics for Protocols

This section contains information that is specific to different protocols.

COM Correlation

The following steps describe how to correlate COM Vuser scripts.

1. Select **View > Output** to display the output tabs at the bottom of the window. Check for errors in the Replay tab. Often, these errors can be corrected by correlation.
2. Select the relevant step in the Step Navigator, and view the step in the **Snapshot pane**.
3. Right click the value in the snapshot and select **Create correlation**. This will open the **Design Studio** window.
4. Select the value you would like to correlate by highlighting it in the grid and clicking the **Correlate** button.

When a value is correlated, VuGen adds the correlation parameter and saves the original value in a comment in the script.

```
252 | /* Correlation comment - Do not change! Original value='1' Name = 'CorrelationParameter' Type = 'Manual' */  
253 | lr_save_rs_param(_Recordset_45,  
254 |     1,  
255 |     2,  
256 |     0,  
257 |     "CorrelationParameter");
```

C Vuser Scripts Correlation Functions

To correlate statements for protocols that do not have specific functions, you can use the C Vuser correlation functions. These functions can be used for all C-type Vusers, to save a string to a parameter and retrieve it when required.

lr_eval_string	Replaces all occurrences of a parameter with its current value.
lr_save_string	Saves a null-terminated string to a parameter.
lr_save_var	Saves a variable length string to a parameter.

For additional information about the syntax of these functions, see the [Function Reference \(Help > Function Reference\)](#).

Using lr_eval_string

In the following example, lr_eval_string replaces the parameter row_cnt with its current value. This value is sent to the Output window using lr_output_message.

```
lrd_stmt(Csr1, "select count(*) from employee", -1, 1 /*Deferred*/, ...);
lrd_bind_col(Csr1, 1, =;COUNT_D1, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_save_col(Csr1, 1, 1, 0, "row_cnt");
lrd_fetch(Csr1, 1, 1, 0, PrintRow2, 0);
lr_output_message("value: %s" , lr_eval_string("The row count is: <row_cnt>"));
```

Using lr_save_string

To save a NULL terminated string to a parameter, use **lr_save_string**. To save a variable length string, use **lr_save_var** and specify the length of the string to save.

In the following example, lr_save_string assigns 777 to a parameter emp_id. This parameter is then used in another query or for further processing.

```
lrd_stmt(Csr1, "select id from employees where name='John'",...);
lrd_bind_col(Csr1,1,=;ID_D1,...);
lrd_exec(Csr1, ...);
lrd_fetch(Csr1, 1, ...);
/* GRID showing returned value "777" */
lr_save_string("777", "emp_id");
```

Database, RTMP and COM Correlation from a Snapshot

The following steps describe how to correlate scripts from a snapshot.

In order to correlate values from a snapshot, you must first select the **Enable correlation from snapshots** option in the **Scripting > Correlation** section of the ["Options Dialog Box" on page 86](#).

Note: When you correlate a value from a snapshot, VuGen may create a boundary-based correlation, even though the recording correlation option is set to use regular expressions.

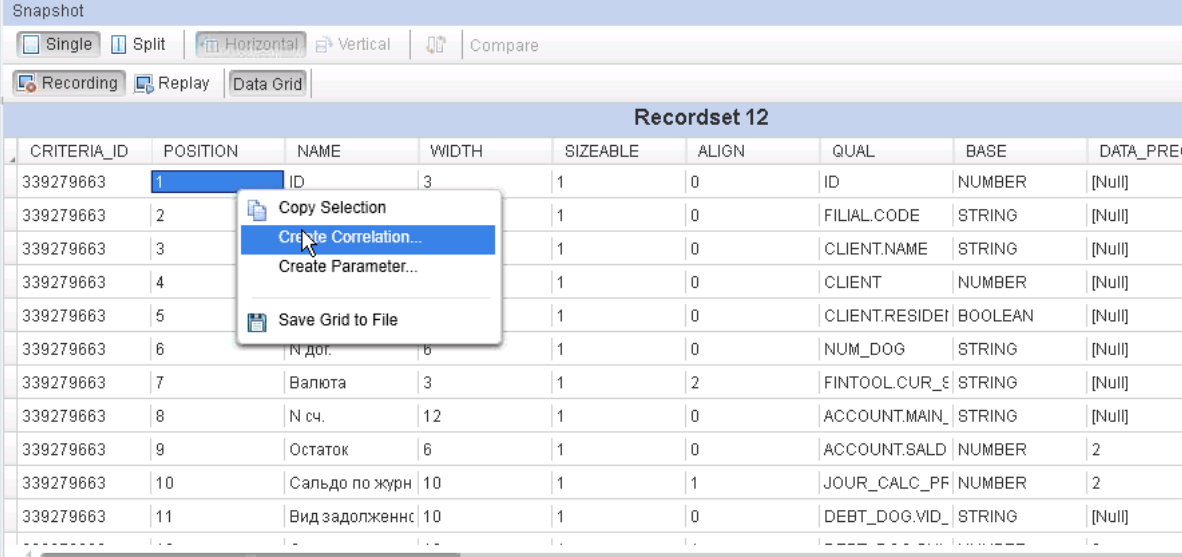
This task applies to the following protocols:

- Database Protocols
- RTMP
- COM Protocols

1. Open the **Output Pane**

Select **View > Output** to display the output tabs at the bottom of the window. Check for errors in the Replay tab. Often, these errors can be corrected by correlation.

2. Select the relevant step in the **Step Navigator**, and view the step in the **Snapshot pane**. Right click the value in the snapshot and select **Create correlation**. This will open the **Design Studio** window.



Snapshot

Single Split Horizontal Vertical Compare

Recording Replay Data Grid

Recordset 12

CRITERIA_ID	POSITION	NAME	WIDTH	SIZEABLE	ALIGN	QUAL	BASE	DATA_PREC
339279663	1	ID	3	1	0	ID	NUMBER	[Null]
339279663	2	FILIAL.CODE	1	0	0	FILIAL.CODE	STRING	[Null]
339279663	3	CLIENT.NAME	1	0	0	CLIENT.NAME	STRING	[Null]
339279663	4	CLIENT	1	0	0	CLIENT	NUMBER	[Null]
339279663	5	CLIENT.RESIDEI	1	0	0	CLIENT.RESIDEI	BOOLEAN	[Null]
339279663	6	NUM_DOG	1	0	0	NUM_DOG	STRING	[Null]
339279663	7	Валюта	3	1	2	FINTOOL.CUR_5	STRING	[Null]
339279663	8	Н сч.	12	1	0	ACCOUNT.MAIN_	STRING	[Null]
339279663	9	Остаток	6	1	0	ACCOUNT.SALD	NUMBER	2
339279663	10	Сальдо по журн	10	1	1	JOUR_CALC_PF	NUMBER	2
339279663	11	Вид задолженнс	10	1	0	DEBT_DOG.VID_	STRING	[Null]

3. You can select the value you would like to correlate by highlighting it in the grid and clicking the **Correlate** button.
4. When a value is correlated, VuGen adds the correlation parameter, and saves the original value in a comment in the script.

```

252  /* Correlation comment - Do not change! Original value='1' Name ='CorrelationParameter' Type ='Manual' */
253  lrc_save_rs_param(_Recordset_45,
254      1,
255      2,
256      0,
257      "CorrelationParameter");

```

Flex (RTMP/AMF) Correlation

This task describes how to correlate Flex (RTMP/AMF) Vuser scripts. For details on how to use XPath correlation in Flex Vuser scripts, see ["Flex XPath Correlation" on page 258](#).

1. **Locate the step in your script that failed due to dynamic values that need correlation.**

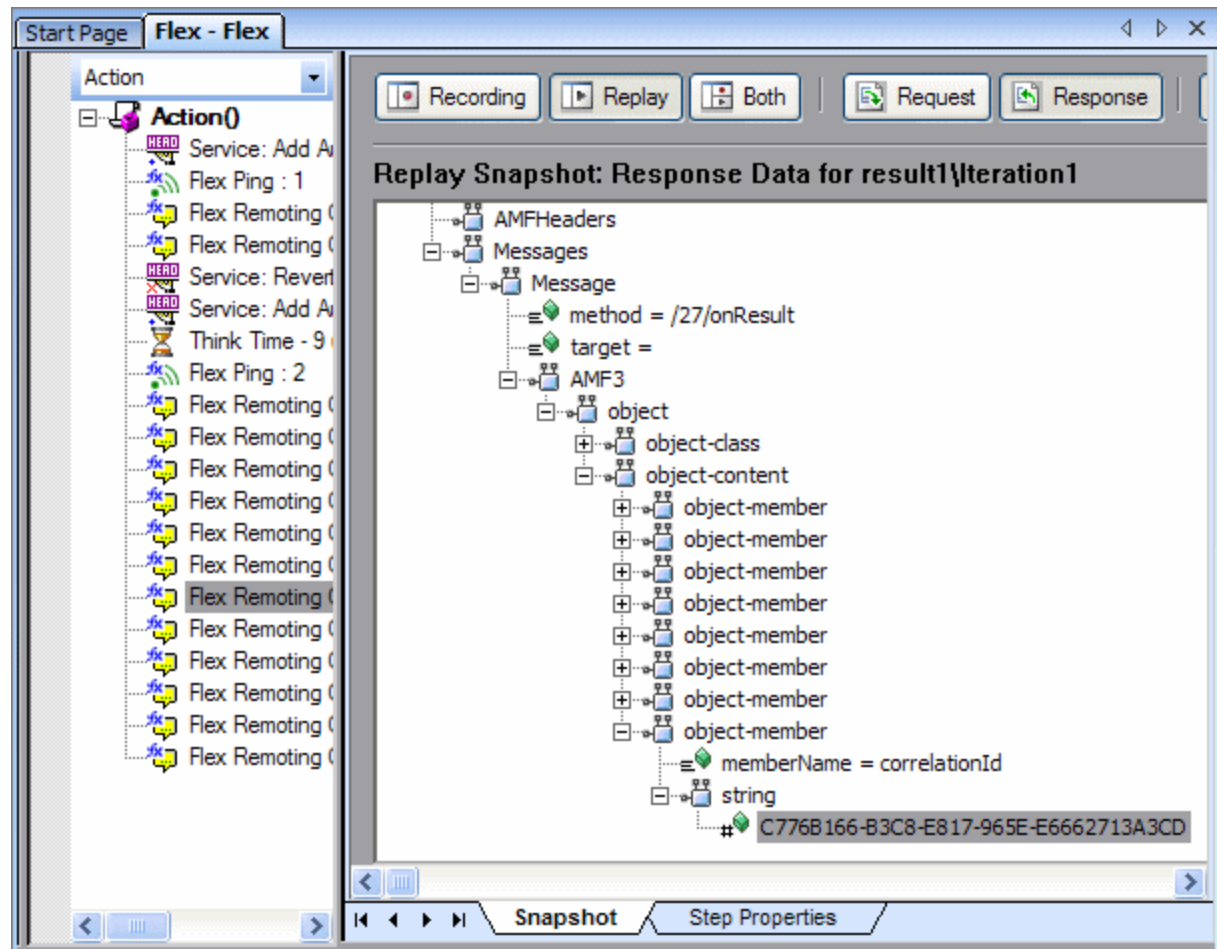
Use the Replay Log to assist you in finding the problematic step. These errors are not always obvious, and you may detect them only by carefully examining Vuser log files.

```
Action.c(16): Error Server returned error for message #1 : "Incorrect session ID sent"/
```

```
Action.c(16): There was an error during the Flex Call ("ConnStatus")
```

2. **Identify the server response with the correct value in one of the previous steps.**

Double-Click the error in the Replay log to go to the step with the error. Examine the preceding steps in the Step Navigator and look for the value in the **Server Response** tab.



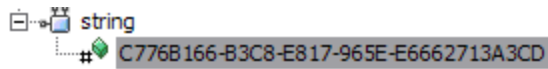
3. **Save the entire server response to a parameter.**

Before you extract the value, the entire server response should be saved to a parameter as follows:

- Right-click the step node (in the left Action pane) corresponding to the server response containing the value and select **Properties**.
- In the Flex (or AMF) Call Properties dialog box, type a **Response parameter** name.
- Click **OK** to save the new parameter name.

4. **Save the original server response value to a parameter.**

- In the XML tree of the Server Response, right-click the node above the value (for example, string), and select **Save value in parameter**.

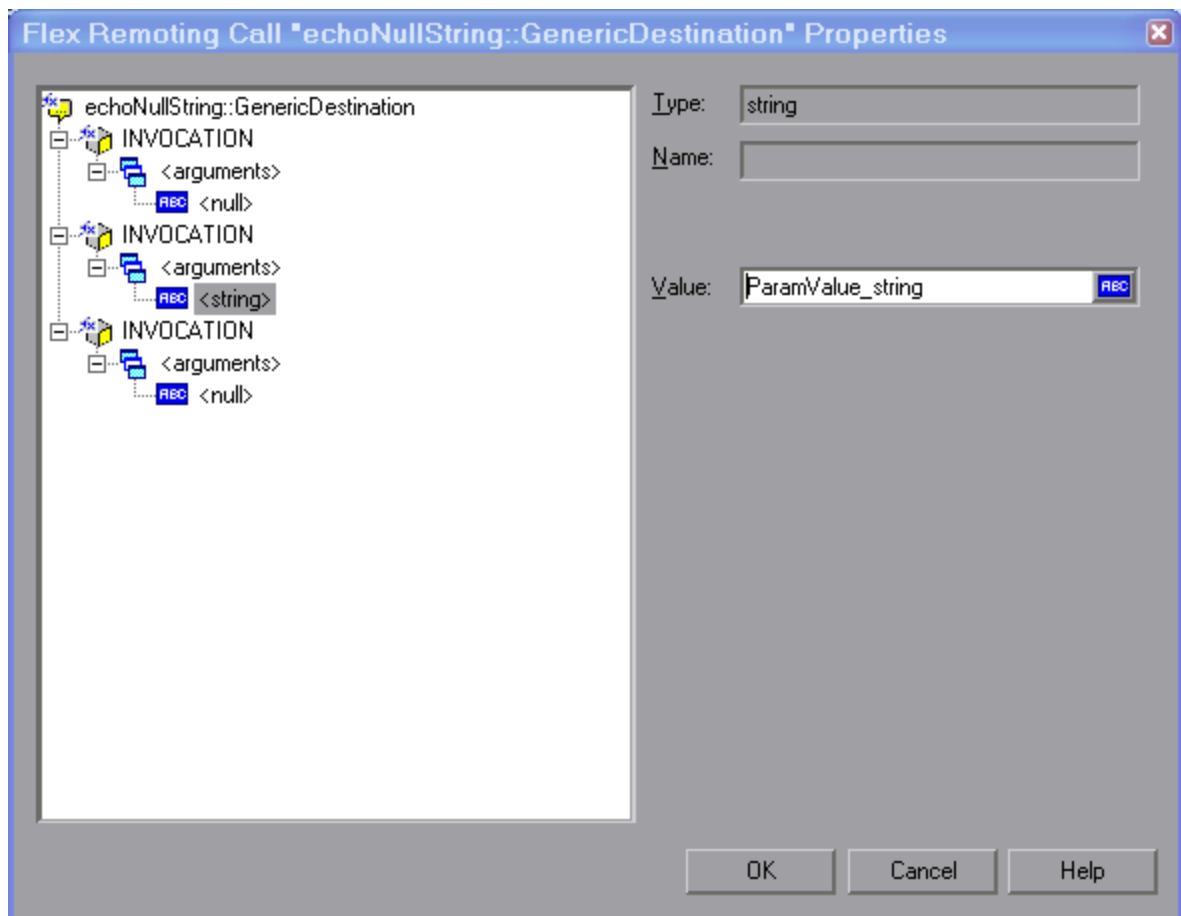


- In the XML Parameter Properties dialog, specify a parameter **Name**. You will use this name in subsequent steps.
- Click **OK**. The script now contains a new function, **lr_xml_get_values**.

5. Insert the parameter in the subsequent calls.

In the VuGen Editor, beginning with the call that failed, replace the value in all subsequent calls to the object with the parameter that you defined:

- Right-click the step node (in the Action pane) corresponding to the failed call and select **Properties**.
- Locate the argument that required correlation.
- In the **Value** box, type the parameter name in curly brackets, for example, **{ParamValue_string}**.



Click **OK**.


6. Run the script.

Make sure that VuGen properly substitutes the argument value with the parameter value that you saved.


Flex XPath Correlation

This topic describes how to use XPath correlation in Flex Vuser scripts. You use the XML View inside the Snapshot pane to perform the correlation. Before you can successfully implement XPath correlation, you must first configure the recording options.

For details on how to use regular correlation in Flex Vuser scripts, see ["Correlate Scripts Using Design Studio" on page 250](#).

1. Configure the recording options
 - a. Select **Record > Recording Options**.
 - b. Under **Flex**, click **Externalizable Objects**.
 - c. Click **Serialize objects using**, and select **Custom Java Classes**.
 - d. Click the **Add jar or zip file** button .
 - e. On the LiveCycle installation discs, locate the following three files, and add them to the **Classpath Entries** list:
 - i. **flex.jar**
 - ii. **flex-messaging-common.jar**
 - iii. **flex-messaging-core.jar**

Ensure that the added files exist in the same location on all load generator machines.

2. Record the Vuser script.
3. In the Editor, click inside the **flex_amf_call** step that contains the data you want to correlate, or in the Step Navigator, double-click the **flex_amf_call** step that contains the data you want to correlate.
4. Click **View > Snapshot** or click the **Snapshot** button  on the VuGen toolbar.
5. In the Snapshot pane, click the **Response Body** tab.
6. On the right-side of the Snapshot pane, click the **XML View** tab.
7. In the XML View, locate and select the entire string that contains the dynamic data that requires correlation.
8. Right-click inside the selection, and select **Create Correlation**. The Design Studio opens. For details on how to use Design Studio, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

When the correlation is complete, VuGen adds a **web_reg_save_parm_xpath** step to the Vuser script.

Java Custom Correlation

You can manipulate script objects like any Java object. One use of this is to write custom correlation code.

Automatic correlation is not supported in Java over HTTP, but you can use custom correlation as described in this example.

Given this business logic class:

```
public class RmiMessage implements Serializable {  
    List<Integer> integers;  
  
    public RmiMessage(List<Integer> myIntegers) {  
        this.integers = myIntegers;  
    }  
  
    public void setIntegers(List<Integer> myIntegers) {  
        this.integers = myIntegers;  
    }  
}
```

And given the following code in Action.java:

```
String _string9 = "com.hpe.lr.testing.rmi.RmiMessage __CURRENT_OBJECT = {"  
    + "java.util.List myIntegers = {"  
        + "super = {"  
            + "super = {"  
            + "}"  
            + "int modCount = #0#"  
        + "}"  
        + "java.lang.Object a[] = {"  
            + "java.lang.Integer a[0] = {"  
+ "super = {"  
+ "}"  
+ "int value = #4#"  
        + "}"  
        + "java.lang.Integer a[1] = {"  
+ "super = {"  
+ "}"  
+ "int value = #21#"
```

```
+ "}"  
+ "}"  
+ "}"  
+ "}";  
  
RmiMessage _message = (com.hpe.lr.testing.rmi.RmiMessage)lr.deserialize(_  
    string9, 2);  
  
_remote.exampleMethodCall(_message);
```

After the call to `lr.deserialize`, you have the business logic object, which can then be manipulated.

In this example, correlation is done by setting the integers list:

```
RmiMessage _message = (com.hpe.lr.testing.rmi.RmiMessage) lr.deserialize(_
    string9, 2);

_message.setIntegers(Arrays.asList(4,5,7)); // call business logic methods
    here to adjust object to your needs

_remote.exampleMethodCall(_message);
```

Java Scripts Correlation - Serialization

In RMI and some cases of CORBA, the client AUT creates a new instance of a Java object using the **java.io.Serializable** interface. It passes this instance as a parameter for a server invocation. In the following segment, the instance **p** is created and passed as a parameter.

```
// AUT code:
java.awt.Point p =
    new java.awt.Point(3,7);
map.set_point(p);
```

The automatic correlation mechanism is ineffective here, since the object did not return from any previous call. In this case, VuGen activates the serialization mechanism and stores the object being passed as a parameter. It saves the information to a binary data file under the user folder. Additional parameters are saved as new binary data files, numbered sequentially. VuGen generates the following code:

```
public class Actions {
    // Public function: init
    public int init() throws Throwable {
        java.awt.Point p =
            (java.awt.Point)lr.deserialize(0, false);
        map.set_point(p);
    }
}
```

```
}
}
```

The integer passed to **lr.deserialize** represents the number of binary data files in the Vuser folder.

To parameterize the recorded value, use the Java setLocation method (for information, see <https://docs.oracle.com/en/java/>). The following example uses the **setLocation** method to set the value of the object, p.

```
public class Actions {
    // Public function: init
    public int init() throws Throwable {
        java.awt.Point p = (java.awt.Point)lr.deserialize(0, false);
        p.setLocation(2,9);
        map.set_point(p);
    }
}
```

In certain instances the public method of **setLocation** is not applicable. As an alternative, you can use the API of your class that incorporate get or set accessor methods. If you are working with AUT classes that do not have get/set methods or use private methods, or if you are unfamiliar with the classes' API, you can use VuGen's built-in serialization mechanism. This mechanism allows you to expand objects in their ASCII representation and manually parameterize the script. You enable this mechanism in the Recording Options dialog box. For details, see ["Recording Properties > Serialization Options - Recording Options" on page 198](#).

VuGen generates an **lr.deserialize** method that deserializes the data or displays complex data structures as serial strings. Once the structure is broken down to its components, it is easier to parameterize. The **lr.deserialize** method receives two arguments, a string and an integer. The string is the parameter's value that is to be substituted during replay. The integer is the index number of binary file to load.

If you choose not to expand objects in your script by clearing the Unfold Serialized Objects check box, you can control the serialization mechanism by passing arguments to the lr.deserialize method. The first argument is an integer indicating the number of binary files to load. The second integer is a boolean value:

true	Use VuGen's serialization mechanism.
false	Use the standard Java serialization mechanism.

The following segment shows a generated script in which the serialization mechanism was enabled.

```
public class Actions {
    // Public function: init
    public int init() throws Throwable {
        _string = "java.awt.Point __CURRENT_OBJECT = {" +
            "int x = "#5#" +
            "int y = "#8#" +
```

```
    "};  
    java.awt.Point p = (java.awt.Point)lr.deserialize(_string,0);  
    map.set_point(p);  
  }  
}
```

The string values are placed between delimiters. The default delimiter is "#". You can change the delimiter in the **Serialization** tab of the recording options. Delimiters are used to speed up the parsing of the string during replay.

When modifying the string, you must maintain the following rules:

- Order of lines may not be changed. The parser reads the values one-by-one—not the member names.
- Only values between two delimiters may be modified.
- Object references may not be modified. Object references are indicated only to maintain internal consistency.
- "_NULL_" can appear as a value, representing the Java null constant. You can replace it with string type values only.
- Objects may be deserialized anywhere in the script. For example, you can deserialize all objects in the **init** method and use the values in the **action** method.
- Maintain internal consistency for the objects. For example, if a member of a vector is **element count** and you add an element, you must modify the element count.

In the following segment, a vector contains two elements:

```
public class Actions {  
  // Public function: init  
  public int init() throws Throwable {  
    _string = "java.util.Vector CURRENTOBJECT = {" +  
      "int capacityIncrement = "#0#" +  
      "int elementCount = #2#" +  
      "java/lang/Object elementData[] = {" +  
        "elementData[0] = #First Element#" +  
        "elementData[1] = #Second Element#" +  
        "elementData[2] = _NULL_" +  
        ....  
        "elementData[9] = _NULL_" +  
      "}" +  
    "};  
    _vector = (java.util.Vector)lr.deserialize(_string,0);  
    map.set_vector(_vector);  
  }  
}
```

In the following example, one of the vector's elements was changed—a "_NULL_" value was changed to "Third element". In coordination with the addition of the new element, the **elementCount** member was modified to **3**.

```
public class Actions {  
    // Public function: init  
    public int init() throws Throwable {  
        _string = "java.util.Vector CURRENTOBJECT = {" +  
            "int capacityIncrement = "#0#" +  
            "int elementCount = #3# " +  
            "java/lang/Object elementData[] = {" +  
                "elementData[0] = #First Element#" +  
                "elementData[1] = #Second Element#" +  
                "elementData[2] = #Third Element#" +  
                ....  
                "elementData[9] = _NULL_" +  
            "}" +  
        "};"  
        _vector = (java.util.Vector)lr.deserialize(_string,0);  
        map.set_vector(_vector);  
    }  
}
```

Due to the complexity of the serialization mechanism, which opens up the objects to ASCII representation, opening large objects while recording may increase the time required for script generation. To decrease this time, you can specify flags which will improve the performance of the serialization mechanism.

When adding **lr.deserialize** to your script, we recommend that you add it to the **init** method—not the **action** method. This will improve performance since VuGen will only deserialize the strings once. If it appears in the **action** method, VuGen will deserialize strings for every iteration.

Microsoft .NET Correlation

This task describes how to correlate Microsoft .NET Vuser scripts.

For primitive values, you should generate the script with output parameter values and examine the output parameters for correlations.

1. Select **Recording > Recording Options**, and select the **General > Script** node.
2. Select the **Insert output parameter values** check box. Click **OK** to close the Recording Options dialog box.
3. Select **Record > Regenerate Script** to regenerate the script.
4. Search the commented output primitive values for correlations.

```
lr.log("Event 104: IEchoer_1.EchoInt16(short.MaxValue);");
Int16RetVal = IEchoer_1.EchoInt16(short.MaxValue);
// Int16RetVal = -32759;

Int16 arg_55;
arg_55 = short.MaxValue;
lr.log("Event 105: IEchoer_1.EchoInt16ByRef(ref arg_55);");
Int16RetVal = IEchoer_1.EchoInt16ByRef(ref arg_55);
// Int16RetVal = -32759;
// arg_55 = 32757;
```

For more information about using correlation functions, see the [Function Reference \(Help > Function Reference\)](#).

Oracle NCA Correlation

The following steps describe different items in Oracle NCA scripts that may need correlation.

Correlate Statements for Load Balancing

VuGen supports load balancing for multiple application servers. You correlate the HTTP return values with the **nca_connect_server** parameters. The Vuser then connects to the relevant server during test execution, applying load balancing. The following steps describe how to correlate statements for load balancing.

1. **Record a multi-protocol script.**

Record a multi-protocol script for Oracle NCA and Web Protocols. Perform the desired actions and save the script.

2. **Define parameters for host and host arguments.**

Define two variables, **serverHost** and **serverArgs**, for parameterization:

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
    "LB=<PARAM name=\"serverHost\" value=\"\", \"RB=\">", LAST);
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
    "LB=<PARAM name=\"serverArgs\" value=\"\", \"RB=\">", LAST);
```

3. **Assign values to serverHost and serverArgs:**

```
web_url("step_name", "URL=http://server1.acme.com/test.htm", LAST);
```

4. **Modify the nca_connect_server statement from:**

```
nca_connect_server("199.203.78.170", 9000/*version=107*/,
    "module=e:\\apps\\nca...fndnam=apps ");
```

to:

```
nca_connect_server("{ serverHost }", "9000"/*version=107*/, "  
{serverArgs}");
```

The script should now look like this:

```
web_set_max_html_param_len("512");  
web_reg_save_param("serverHost", "NOTFOUND=ERROR",  
    "LB=<PARAM name=\"serverHost\" value=\"\", \"RB=\">", LAST);  
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",  
    "LB=<PARAM name=\"serverArgs\" value=\"\", \"RB=\">", LAST);  
web_url("step_name", "URL=http://server1.acme/test.htm", LAST);  
nca_connect_server("{serverHost}", "9000"/*version=107*/, "{serverArgs}");
```

Correlate the icx_ticket Variable

The `icx_ticket` variable, is part of the information sent in the `web_url` and `nca_connect_server` functions:

```
web_url("fnd_icx_launch.runforms",  
    "URL=http://ABC-123:8002/pls/VIS/fnd_icx_launch.runforms\?ICX_  
TICKET=5843A55058947ED3=;RESP_APP=AR=;RESP_KEY=RECEIVABLES_MANAGER=;SECRP_  
KEY=STANDARD", LAST);
```

This **icx_ticket** value is different for each recording. It contains cookie information sent by the client. To correlate your recording, add `web_reg_save_param` before the first occurrence of the recorded **icx_ticket** value:

```
web_reg_save_param("icx_ticket", "LB=TICKET=", "RB==;RES", LAST);  
...  
web_url("fnd_icx_launch.runforms",  
    "URL=http://ABC-123:8002/pls/VIS/fnd_icx_launch.runforms\?ICX_TICKET={icx_ticket}=;RESP_  
APP=AR=;RESP_KEY=RECEIVABLES_MANAGER=;SECRP_KEY=STANDARD", LAST);
```

Note: The left and right boundaries of `web_reg_save_param` may differ depending on your application setup.

Correlate the JServSessionIdroot Values

The **JServSessionIdroot** value is a cookie that the application sets to store the session ID. In most cases, VuGen automatically correlates this value and inserts a `web_reg_save_param` function. If VuGen did not add this function automatically, you add it manually, replacing all of its occurrences with the parameter name.

To identify the value that you need to correlate, open the Execution log (**View > Output Window**) and locate the response body.

```
vuser_init.c(8):      Set-Cookie: JServSessionIdroot=my1sanw2n1.JS4; path=/\r\n
```

```
vuser_init.c(8):      Content-Length: 79\r\n
vuser_init.c(8):      Content-Type: text/plain\r\n
vuser_init.c(8):      \r\n
vuser_init.c(8):      81-byte response body for "http://ABC-
123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo=;
                        ifhost=mercury;ifip=123.45.789.12" (RelFrameId=1)
vuser_init.c(8):
/servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdroot=my1sanw2n1.JS4\r\n
```

To correlate this dynamic value, insert a **web_reg_save_param** function before the first occurrence and then replace the variable value with the parameter name throughout the script. In this example, the right and left boundaries are `\r` and `\n`, but you should check your specific environment to determine the exact boundaries in your environment.

```
web_reg_save_param("NCAJServSessionId","LB=\r\n\r\n","RB=\r","ORD=1",LAST);
web_url("f60servlet",
        "URL= http://ABC-
        "123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo=;
        "ifhost=mercury;ifip=123.45.789.12", LAST);
web_url("oracle.forms.servlet.ListenerSer",
        "URL=http://ABC-123{NCAJServSessionId}?ifcmd=getinfo=;
        "ifhost=mercury;ifip=123.45.789.12", LAST);
```

Siebel Correlation

The following steps describe how to correlate Siebel Web Vuser scripts.

Correlation Library

To assist you with correlation, Siebel has released a correlation library file as part of the Siebel Application Server version 7.7. This library is available only through Siebel. The library file, **ssdtcorr.dll**, is located under the `siebsrvr\bin` folder for Windows and under `siebsrvr/lib` for Linux installations.

The library file, **ssdtcorr.dll**, must be available to all machines where a Load Generator or Controller reside. The following steps describe how to enable correlation with this library.

1. Copy the DLL file into the bin folder of the product installation.
2. Open a multi-protocol script using the **Siebel-Web** Vuser type.
3. Enable UTF-8 support in the **Recording Options > HTTP Properties > Advanced** node.
4. Open the recording option's Correlation node and click **Import**. Import the rules file, **WebSiebel77Correlation.cor**, from the `\dat\webrulesdefaultsetting` folder. If you are prompted with warnings, click **Override**.

To revert to the default correlation, delete all of the Siebel rules and click **Use Defaults**.

When using the Siebel correlation library, verify that the SWE count rules (where the left boundary contains the **SWEC** string) are not disabled.

Correlation Rules

VuGen's native built-in rules for the Siebel server detect the Siebel server variables and strings, automatically saving them for use at a later point within the script. The rules list the boundary criteria that are unique for Siebel server strings.

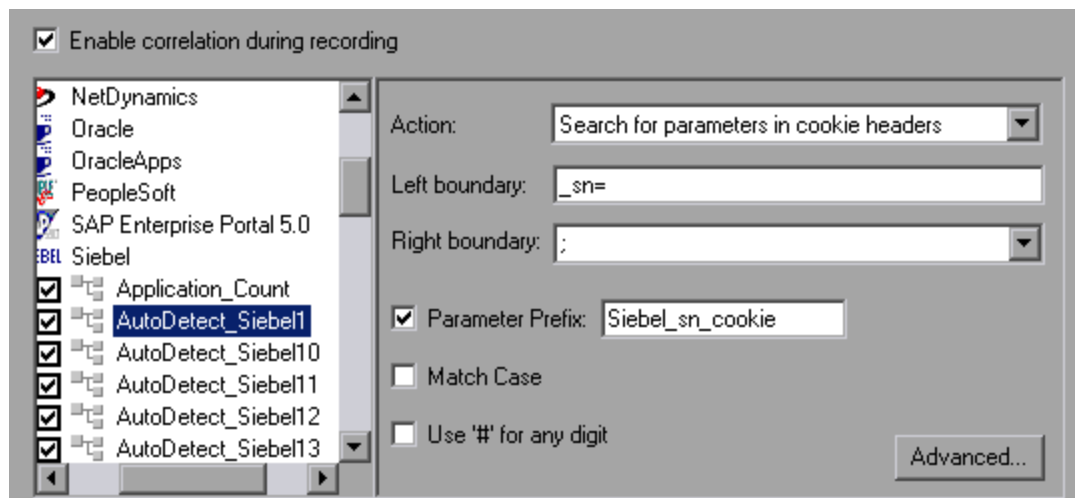
When VuGen detects a match using the boundary criteria, it saves the value between the boundaries to a parameter. The value can be a simple variable or a public function.

In normal situations, you do not need to disable any rules. In some cases, however, you may want to disable rules that do not apply. For example, disable Japanese content check rules when testing English-only applications.

Another reason to disable a rule is if the Controller explicitly requires an error condition to be generated. View the rule properties in the recording options and determine the conditions necessary for your application.

Simple Variable Correlation

In the following example, the left boundary criteria is `_sn=`. For every instance of `_sn=` in the left boundary and `;` in the right, VuGen creates a parameter with the **Siebel_sn_cookie** prefix.



In the following example, VuGen detected the `_sn` boundary. It saved the parameter to `Siebel_sn_cookie6` and used it in the `web_url` function.

```
/* Registering parameter(s) from source
web_reg_save_param("Siebel_sn_cookie6",
"LB/IC=_sn=",
"RB/IC=";",
"Ord=1",
"Search=headers",
"RelFrameId=1",
LAST);
...
```

```
web_url("start.swe_3",
"URL=http://cannon.hplab.com/callcenter_enu/start.swe?SWECmd=GotoPostedAction;;SWEDIC=true;;_sn={Siebel_sn_cookie6};;SWEC={Siebel_SWECCount};;SWEFrame=top._sweclient;;SWECs=true",
"TargetFrame=",
"Resource=0",
"RecContentType=text/html",
"Referer=http://cannon.hplab.com/callcenter_enu/start.swe?SWECmd=GetCachedFrame;;_sn={Siebel_sn_cookie6};;SWEC={Siebel_SWECCount};;SWEFrame=top._swe",
"Snapshot=t4.inf",
"Mode=HTML",
LAST);
```

Function Correlation

In certain instances, the boundary match is a function. Functions generally use an array to store the runtime values. In order to correlate these values, VuGen parses the array and saves each argument to a separate parameter using the following format:

```
<parameter_name> = <recorded_value> (display_name)
```

The display name is the text that appears next to the value, in the Siebel Application.

VuGen inserts a comment block with all of the parameter definitions.

```
/* Registering parameter(s) from source task id 159
// {Siebel_Star_Array_Op33_7} = ""
// {Siebel_Star_Array_Op33_6} = "1-231"
// {Siebel_Star_Array_Op33_2} = ""
// {Siebel_Star_Array_Op33_8} = "Opportunity"
// {Siebel_Star_Array_Op33_5} = "06/26/2003 19:55:23"
// {Siebel_Star_Array_Op33_4} = "06/26/2003 19:55:23"
// {Siebel_Star_Array_Op33_3} = ""
// {Siebel_Star_Array_Op33_1} = "test camp"
// {Siebel_Star_Array_Op33_9} = ""
// {Siebel_Star_Array_Op33_rowid} = "1-6F"
// */
```

In addition, when encountering a function, VuGen generates a new parameter for **web_reg_save_param**, **AutoCorrelationFunction**. VuGen also determines the prefix of the parameters and uses it as the parameter name. In the following example, the prefix is **Siebel_Star_Array_Op33**.

```
web_reg_save_param("Siebel_Star_Array_Op33",
"LB/IC=`v`",
"RB/IC=`",
"Ord=1",
"Search=Body",
```

```
"RelFrameId=1",  
"AutoCorrelationFunction=flCorrelationCallbackParseStarArray",  
LAST);
```

VuGen uses the parameters at a later point within the script. In the following example, the parameter is called in **web_submit_data**.



Example: web_submit_data("start.swe_14", "Action=http://cannon.hplab.com/callcenter_enu/start.swe", "Method=POST", "RecContentType=text/html", "Referer=", "Snapshot=t15.inf", "Mode=HTML", ITEMDATA, "Name=SWECLK", "Value=1", ENDITEM, "Name=SWEField", "Value=s_2_1_13_0", ENDITEM, "Name=SWER", "Value=0", ENDITEM, "Name=SWESP", "Value=false", ENDITEM, "Name=s_2_2_29_0", "Value={Siebel_Star_Array_Op33_1}", ENDITEM, "Name=s_2_2_30_0", "Value={Siebel_Star_Array_Op33_2}", ENDITEM, "Name=s_2_2_36_0", "Value={Siebel_Star_Array_Op33_3}", ENDITEM, ...

During replay, Vusers do a callback to the public function, using the array elements that were saved as parameters.



Note: Correlation for the **SWEC** parameter is not done through the correlation rules. VuGen handles it automatically with a built-in detection mechanism. For more information, see below.

SWEC Correlation

SWEC is a parameter used by Siebel servers representing the number of user clicks. The SWEC parameter usually appears as an argument of a URL or a POST statement. For example:

```
GET /callcenter_enu/start.swe?SWECmd=GetCachedFrame=;_sn=2-  
mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_=;SWEC=1;SWEFrame=top._swe._sweapp  
HTTP/1.1
```

or

```
POST /callcenter_enu/start.swe HTTP/1.1  
...  
\r\n\r\n  
SWERPC=1; SWEC=0;_sn=2-mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_  
=;SWECmd=InvokeMethod...
```

VuGen handles the changes of the SWEC by incrementing a counter before each relevant step. VuGen stores the current value of the SWEC in a separate variable (Siebel_SWECCount_var). Before each step, VuGen saves the counter's value to a VuGen parameter (Siebel_SWECCount).

In the following example, web_submit_data uses the dynamic value of the SWEC parameter, Siebel_SWECCount.

```
Siebel_SWECCount_var += 1;
lr_save_int(Siebel_SWECCount_var, "Siebel_SWECCount");
web_submit_data("start.swe_8",
    "Action=http://cannon.hplab.com/callcenter_enu/start.swe",
    "Method=POST",
    "TargetFrame=",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t9.inf",
    "Mode=HTML",
    "EncodeAtSign=YES",
    ITEMDATA,
    "Name=SWERPC", "Value=1", ENDITEM,
    "Name=SWEC", "Value={Siebel_SWECCount}", ENDITEM,
    "Name=SWECmd", "Value=InvokeMethod", ENDITEM,
    "Name=SWEService", "Value=SWE Command Manager", ENDITEM,
    "Name=SWEMethod", "Value=BatchCanInvoke", ENDITEM,
    "Name=SWEIPS", ...
    LAST);
```

Note that the SWEC parameter may also appear in the referrer URL. However, its value in the referrer URL usually differs from its value in the requested URL. VuGen handles this automatically.

Correlate SWECCount Parameters

The SWECCount parameter value is usually a small number consisting of one or two digits. It is often difficult to determine where to replace the recorded value with a parameter.

In the **web_submit_data** function, VuGen only replaces it in the SWEC field.

In URLs, VuGen only replaces the value when it appears after the strings "SWEC=" or "SWEC`".

The parameter name for all the SWECCount correlations is the same.

Correlate ROWID Parameters

In certain cases, the **rowid** is preceded by its length, encoded in hexadecimal format. Since this length can change, this value must be correlated.

For example, the following string is comprised of a length value and RowID, xxx6_1-4ABCyyy, where 6 is the length, and 1-4ABC is the RowID.

If you define parameters to correlate the string as

```
xxx{rowid_Length}_{rowid}yyy
```

then using this enhanced correlation, VuGen generates the following function before the string:

```
web_save_param_length("rowid", LAST);
```

This function gets the value of **rowid**, and saves its length into the parameter **rowid_length** in hexadecimal format.

Correlate SWET (timestamp) Parameters

The SWETS value in the script, is the number of milliseconds since midnight January 1st, 1970.

VuGen replaces all non-empty timestamps in the script, with the parameter {SiebelTimeStamp}. Before saving a value to this parameter, VuGen generates the following function:

```
web_save_timestamp_param("SiebelTimeStamp", LAST);
```

This function saves the current timestamp to the **SiebelTimeStamp** parameter.

Web Manual Correlations

LoadRunner allows users to manually to add correlations that were missed by the automatic engine; or, alternatively, can be used by scripters that want to identify all correlations by themselves.

(a) Correlate selection

This method allows users to select the desired value (text string) in the script code (function arguments) and requests LoadRunner to check whether this value appears to be a correlation by activating the “Correlate selection” option from context menu (right click à correlate selection).

The logic used here is identical to the “Record-based correlations” mentioned above.

(b) Correlate from snapshot

The user should select a dynamic value from the Server response within a snapshot, and the engine will scan the entire script and replace all relevant occurrences.

(c) Double recording

This method is a design approach that directs the user to record the same business process twice, compare the scripts and locate any mismatches between the two as candidates for the correlation process that will be done manually.

As mentioned before, in order to correlate dynamic values, after locating the appropriate server response, we need to extract this data. The value can be extracted using different methods.

To do this, we use a set of API functions.

The four main correlation APIs are:

1. **web_reg_save_param_ex** – Registers a request to save dynamic data located between specified boundaries.
2. **web_reg_save_param_regexp** – Registers a request to save dynamic data that matches a regular expression.
3. **web_reg_save_param_xpath** – Registers a request to save dynamic data from XML response using XPath expression.

4. `web_reg_save_param_json` – Registers a request to save dynamic data from response that has been formatted as JSON.

Web Manual Correlation in Code

This task describes how to correlate web scripts manually by modifying the code.

1. Locate the string and its details

Identify the statement that contains dynamic data and the patterns that characterize the locations of the data. These patterns may be boundaries or xpaths.

- a. **Identify Patterns using Boundaries**

Use these guidelines to determine and set the boundaries of the dynamic data:

- Analyze the location of the dynamic data within the HTTP response.
- Identify the string that is immediately to the left of the dynamic data. This string defines the left boundary of the dynamic data.
- Identify the string that is immediately to the right of the dynamic data. This string defines the right boundary of the dynamic data.
- The right and left boundaries should be as unique as possible to better locate the strings.
- **`web_reg_save_param_ex`** looks for the characters between (but not including) the specified boundaries and saves the information beginning one byte after the left boundary and ending one byte before the right boundary. **`web_reg_save_param_ex`** does not support embedded boundary characters.

For example, if the input buffer is `{a{b{c}` and `"{"` is specified as a left boundary, and `"}"` as a right boundary, the first instance is `c` and there are no further instances—it found the right and left boundaries but it does not allow embedded boundaries, so `"c"` is the only valid match.

By default, the maximum length of any boundary string is 256 characters. Include a **`web_set_max_html_param_len`** function in your script to increase the maximum permitted length. For example, the following function increases the maximum length to 1024 characters:

These length restrictions do not apply where either the left or right boundaries are blank.

- b. **Identify Patterns using XPath or JSON Path expressions**

Use the snapshot pane to manually search for the XPath of the desired string.

By default, the maximum length of any boundary string is 256 characters. Include a **`web_set_max_html_param_len`** function in your script to increase the maximum permitted length. For example, the following function increases the maximum length to 1024 characters:

These length restrictions do not apply where either the left or right boundaries are blank.

2. Add `web_reg_save_param_*` function

Add a **`web_reg_save_param_ex`** or **`web_reg_save_param_xpath`** function into the script before the statement that contains the dynamic data.

- a. **`web_reg_save_param_ex`**

This function searches server responses in web steps for the left boundary following by the string and the right boundary and saves the string to a parameter named in the function's argument. After finding the specified number of occurrences, **web_reg_save_param_ex** does not search any more responses. For more information, see the Function Reference (**Help > Function Reference**).

b. **web_reg_save_param_xpath**

This function searches server responses in web steps for a specified XPath. The string located in specified XPath is saved to a parameter named in the function's argument. For more information, see the Function Reference (**Help > Function Reference**).

c. **web_reg_save_param_json**

This function searches server responses in Web steps for a specified JSONPath. The located string is saved to a parameter named in the function's argument. For more information, see the Function Reference and Internet resources for JSONPath, XPath for JSON.

3. Replace data with parameter

Select **Edit > Replace** from the VuGen main window to display the Search and Replace dialog box. Search the entire script for the dynamic data and replace it with a parameter. Give the parameter any name and enclose it with braces: {param_name}. You can include a maximum of 64 parameters per script.

Web-based Correlation Using Design Studio

This topic describes how to use the Correlation tab of the Design Studio to correlate Vuser scripts.

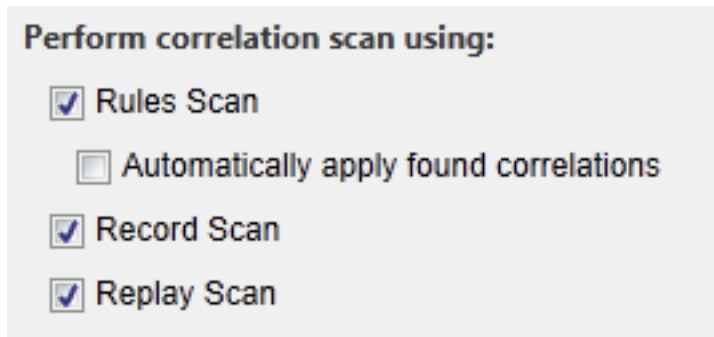
Prerequisites

1. Record a script using one of the following protocols:
 - Web HTTP/HTML
 - Flex
 - RTMP/RTMPT
 - Citrix
 - SAP - Web
 - Oracle NCA

Note: You can only correlate the Web HTTP/HTML steps within your Oracle NCA script and the Web HTTP/HTML protocol must be active. To activate, select **Recording Options > Protocol > Active Protocols > Web HTTP/HTML**.

For additional information on manual correlation, see [How to Correlate Scripts - Oracle NCA](#)

2. Verify that **Record > Recording Options > Correlations > Configuration** has all scan types enabled.



Using the Correlation tab

1. Click the **Design Studio** button to perform an initial scan. This will open the Design Studio dialog box which will scan for response (or record based) correlations and apply correlation rules. The progress bar in the dialog box indicates if the initial scan was successful.

For details on the Correlation tab, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

Note: If no dynamic values for correlation are found, a warning is displayed, advising you to check your recording options. Verify that **Record > Recording Options > Correlations > Configuration** has all scan types enabled.

2. Select which values to correlate by highlighting the value in the grid and clicking the **Correlate** button.

When a value is correlated, VuGen adds a **web_reg_save_param_*** function, and saves the original value in a comment in the script.

You can examine the details of the correlation by expanding the Details Chevron in the dialog box. For details, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

For details on the Correlation tab, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

3. Click the **Add as Rule** button to add a rule type. In addition, you can click the **Edit rule** button to view and edit the corresponding rule if the dynamic value was correlated by a rule. For details, see ["Correlations > Rules Recording Options" on page 151](#).
4. Click the **Replay and Scan** button. The Correlation tab progress bar indicates if additional correlations have been found. Again, you can select which values you would like to correlate by highlighting the value in the grid and clicking the **Correlate** button. You may need to repeat this step several times.
5. When Design Studio no longer finds new correlations, the progress bar will display **Design Studio Scan Complete**.



Tip: If Design Studio does not resolve all correlation-based errors, try to resolve them using manual correlation. For details, see ["Manually Correlate Scripts" on page 242](#).

Winsock Correlation

VuGen's Design Studio provides a user interface for correlating Vuser scripts. Correlation is required when working with dynamic data. A common issue with Winsock Vuser scripts is dynamic ports—ports whose numbers are assigned dynamically. While certain applications always use the same port, others use the next available port. If you try to replay a script and the recorded port is no longer available, your script will fail to replay. To overcome this issue, you must perform correlation—save the actual runtime values and use them within the script.

VuGen uses **lrs_save_param** and **lrs_save_searched_string** functions correlate Winsock scripts. This means that it stores the data that is received for use in a later point within the script. Since correlation stores the received data, it applies only to Receive buffers and not to Send buffers. The recommended procedure is to select a string of dynamic data within the Receive buffer that you want to correlate. Use that same parameter in a subsequent Send buffer.

Correlating a Winsock script

You use the Snapshot pane to begin correlating Winsock Vuser scripts. Both the Text and the Hex tabs in the Snapshot pane have the correlating functionality.

1. In the Snapshot pane, select the data that you want to correlate.
2. Right-click in the selection, and select **Create Correlation** or **Create Boundary Correlation**. The Design Studio opens and displays the Correlation tab.
Note that you can click the number of occurrences in the **Replace/Found** column to enable you to choose the exact occurrences that you want to correlate.
3. Click the **Details** bar to display details about the correlation.
4. Make sure that the **Original Snapshot Step** tab is visible. Notice that the type is either **Data Range** or **Boundary Based**.
5. Click **Correlate** to perform the correlation of the Vuser script.
6. Click **Close** to close the Design Studio. Notice that VuGen has inserted the appropriate correlation functions and comments into the script.

For further details on how to use the Design Studio, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

Parameterization versus Correlation

This type of correlation should not be confused with simple parameterization. Simple parameterization (**Design > Parameters > Create New Parameter**) applies only to data within Send buffers. You set up a parameter and assign it several values. VuGen uses the different values in each of the script runs or iterations.

For details on how to manually correlate a Winsock Vuser script, see ["Winsock Manual Correlation" on the next page](#).

Winsock Manual Correlation

This topic describes how to use the Editor to manually correlate values in a Winsock Vuser script. It is recommended that you first try to correlate values through the Design Studio. For details, see ["Winsock Correlation" on the previous page](#).

In the following example, a user performed a telnet session. The user used a ps command to determine the process ID (PID), and killed an application based on that PID.



Example:

```
frodo:/u/user1>ps
  PID TTY          TIME CMD
 14602 pts/18        0:00 clock
 14569 pts/18        0:03 tcsh
frodo:/u/user1>kill 14602
[3]   Exit 1                  clock
frodo:/u/user1>
```

During execution, the PID of the procedure is different, so killing the recorded PID will be ineffective. To overcome this problem, use **lrs_save_param_ex** to save the current PID to a parameter. Replace the constant PID value with the parameter.

To manually correlate the value:

1. View the buffer contents by selecting **data.ws** in the **Extra File** node of the script. Locate the data that you want to replace with the contents of the saved buffer. Use the right-click menu item **Replace with Parameter**, to replace all instances of the value with the parameter.
2. In the **data.ws** file, note the buffer in which the data was received, for example buf47.



Example:

```
recv buf47 98
  "\r"
  "\x00"
  "\r\n"
  "  PID TTY          TIME CMD\r\n"
  " 14602 pts/18        0:00 clock\r\n"
  " 14569 pts/18        0:02 tcsh\r\n"
  "frodo:/u/lab>"
.
.
.
send buf58
  "kill 14602"
```

3. Determine the offset (starting point) and length of the data string to save. In the above example, the offset of the **PID** is 11 and its length is 5 bytes. For additional information about displaying the data, see ["Data Buffers" on page 828](#).
4. In the recorded section, determine the socket used by buf47. In this example, buf47 used socket1.

```
lrs_receive("socket1", "buf47", LrsLastArg);
```

5. Using the Steps Toolbox, insert an **lrs_save_param_ex** function in the recorded section, after the **lrs_receive** for the relevant buffer. In our example, the buffer is **buf47**. Use the parameter name that you used in Step 1.

```
lrs_save_param_ex("socket1", "user", buf47, 11, 5, ascii, param1);
```

6. Print the parameter to the output using **lr_output_message**.

```
lr_output_message ("param1: %s", lr_eval_string("<param1>"));
```

Replaying

The **Replaying** section describes the various methods that are available to replay Vuser scripts.

Replay Overview

Developing a Vuser script includes the steps shown below. This topic provides an overview of the fourth step, replaying a Vuser script.



After recording a Vuser script, you use VuGen to replay the script. This helps to test the basic functionality of the Vuser script and to uncover errors and issues that need to be addressed. For example, you may find the need for correlation, which is a typical issue that is revealed when you first replay a script.

When the replay is successful, you are ready to enhance the script by adding load-testing functionality to the script. Such functionality could include parameterization, transactions, and rendezvous points.

See also:

- ["Correlation Overview" on page 232](#)
- ["Replay a Vuser Script" on the next page](#)
- ["Debugging Overview" on page 313](#)
- ["Files Generated During Replay" on page 310](#)
- ["Bookmarks Overview" on page 308](#) to learn how to use bookmarks to navigate between sections of the script

- ["Run a Vuser Script from a Command Prompt" on page 308](#)
- ["Run a Vuser Script from a Linux Command Line" on page 849](#)

Replay a Vuser Script

This task describes how to replay a Vuser script.

1. Configure the runtime settings and replay options

- a. Runtime settings control how your Vuser script is replayed. Access the desired runtime settings by double-clicking the **Runtime Settings** node in the **Solution Explorer**.

For an overview of runtime settings, see ["Runtime Settings Overview" on page 283](#).




- b. Specify replay options by selecting **Tools > Options**. For details on options, see ["Options Dialog Box" on page 86](#).

2. Replay the script

To run a Vuser script until the end of the script or until the next breakpoint, perform one of the following:

- Select **Replay > Run**.
- Click the **Run** button  on the Vuser toolbar.
- Press **F5**.

Note: The status of the Vuser script execution appears in the lower left corner of VuGen. The script execution status may be **Ready**, **Running**, or **Paused**.

- To stop a script that is running, click the **Stop Replay**  button on the VuGen toolbar.
- To pause a script that is running, click the **Pause**  button on the VuGen toolbar.
- To continue running a script that is paused, click the **Continue**  button on the VuGen toolbar.

3. View the logs for detailed information

You can view detailed information about how your script behaved during the replay. This information appears in the Output window. For details, see ["Output Pane" on page 77](#).

To learn more about replaying a Vuser script, see ["Replay Overview" on the previous page](#).

Check Linux Compatibility

VuGen provides a tool to check the compatibility of your script to run on Linux-based load generators. You can use this tool to check Linux compatibility while developing the script in VuGen, and so avoid

errors and issues later.

To use the tool, open the script in VuGen, then select **Replay > Test <script name> for Linux Compatibility**.

The tool first checks if the script protocol is supported on Linux; if it is, the tool runs script validation and displays one of the following results:

- **Success** - Linux compatibility test finished with no issues.
- **Warning** - the script might not run correctly on Linux.
- **Error** - the script is blocked, unable to run at all on Linux.

The warning and error messages include a list of found issues.

To see which protocols are supported to run on Linux, fully or partially, see the [System Requirements](#) (check for protocols that run on any supported operating system—this includes Linux systems).



Note: The BinaryXML DFE extension is not supported on Linux. However, the Linux compatibility test may not give a warning if a DFE API call that uses BinaryXml DFE extension was manually added to the script.

Work with Snapshots

This topic describes how to use the basic Snapshot pane functionality.

Show the Snapshot pane

Do one of the following:

- Select **View > Snapshot**.
- Click the **Snapshot** button  on the VuGen toolbar.
- In the Editor, click inside a step that contains a reference to a snapshot.
- In the Step Navigator, double-click a step that contains a reference to a snapshot. Note that in the Step Navigator, each step that contains a snapshot displays a Snapshot icon . You can place your mouse cursor over the snapshot icon to see a thumbnail view of the snapshot.



For more details on the Snapshot pane, see ["Snapshot Pane" on page 62](#).

Copy a snapshot to the clipboard

1. Display the snapshot in the Snapshot pane.
2. Right-click on the snapshot, and then select **Copy Image to the Clipboard**.

Note: The "copy snapshot to the clipboard" functionality is available for only RDP, Citrix, and SAP Vuser scripts.

Copy snapshot text to the clipboard

1. Display the snapshot in the Snapshot pane.
2. Select the text that you want to copy.
3. Right-click in the selected text, and select **Copy Selection**.

Activate the snapshot-on-error functionality

1. Click **Replay > Runtime Settings**. The runtime settings dialog box opens.
2. Under **General**, click **Miscellaneous**.
3. Under **Error Handling**, select the **Generate snapshot on error** check box.

Set the snapshot options

1. Click **Tools > Options**. The Options dialog box opens.
2. Click **Scripting**, and then click **Snapshot**. The snapshot options appear on the right of the dialog box.

Troubleshooting Snapshots

If you encounter a step without a snapshot, follow these guidelines to determine why it is not available. Note that not all steps are associated with snapshots—only steps with screen operations or for Web, showing browser window content, have snapshots.

Several protocols allow you to disable the capturing of snapshots during recording using the Recording options.

If there is no **Record** snapshot displayed for the selected step, it may be due to one of the following:

- The script was recorded with a VuGen version 6.02 or earlier.
- Snapshots are not generated for certain types of steps.
- The imported actions do not contain snapshots.

If there is no **Replay** snapshot displayed for the selected step, it may be due to one of the following:

- The script was recorded with VuGen version 6.02 or earlier.
- The imported actions do not contain snapshots.
- The Vuser files are stored in a read-only folder, and VuGen could not save the replay snapshots.
- The step represents navigation to a resource.



See also:

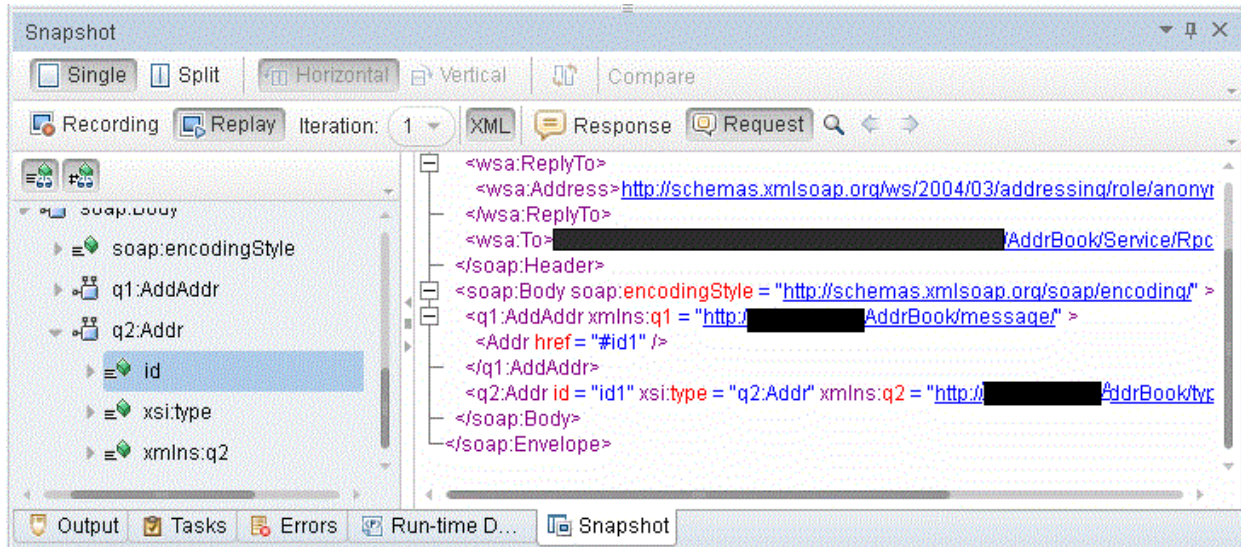
- ["Snapshot Pane" on page 62](#)

Snapshots that Have an XML View


VuGen's Snapshot pane shows various snapshots that were recorded while a Vuser script was recorded or replayed. For specific Vuser protocols only, the Snapshot pane can show XML views of the snapshot. The XML or JSON data is displayed in the XML View tab.

The XML View tab only appears within the **Response** and **Request** Body tabs. The XML view includes its own set of controls and functionality.

The XML view is divided into two areas. On the left is the tree view of the snapshot data, and on the right is the text view.



The two XML views are synchronized. If you select an entry in the tree view, VuGen highlights the corresponding element, attribute, or value in the text view. Alternatively, if you double-click an element, attribute, or value in the text view, VuGen opens the tree view as required, and highlights the corresponding entry.


The splitter controls () between the tree view and text view enable you to set the proportion of the available space that is occupied by each of the views.

Controls to the left of the text view enable you to expand and collapse elements within the text view. Click the right-facing arrow to expand an element in the view, and click the down-facing arrow to collapse an element in the view. Note that you can place the mouse cursor inside an element in the text view and then press the <+> key to expand the element or the <-> key to collapse the element, as appropriate.

After recording a Vuser script, you can use the XML View in the Snapshot pane to add a text check to the script. For details, see ["Add a Text Check From the XML View in the Snapshot Pane" below](#).

Add a Text Check From the XML View in the Snapshot Pane

After recording a Vuser script, you can add a text check from the XML view in the Snapshot pane.

1. Click **View > Snapshot**, or click the **Show Snapshot Pane** button  on the VuGen toolbar.
2. In the Snapshot pane, display a snapshot that contains the text that you want to verify.
3. On the right-side of the Snapshot pane, click the **XML View** tab.
4. In the Snapshot pane, click the **Response Body** tab.
5. In either the Tree view or the Grid view, locate and select the text string that you want to verify.
6. Right-click inside the selection, and select **Add Text Check Step**. The Find Text dialog box opens.
7. Modify the options in the Find Text dialog box. For details on the dialog box options, press F1

when in the dialog box to open the Function Reference.

8. Click **OK** to insert a **web_reg_find** step into the Vuser script.

See also:

- ["Snapshots that Have an XML View" on page 281](#)

Running a Vuser as a Process or Thread

The Controller uses a driver program (such as *mdrv.exe* or *r3vuser.exe*) to run your Vusers.

If you run each Vuser as a **process**, then the same driver program is launched (and loaded) into the memory again and again for every instance of the Vuser. Loading the same driver program into memory uses up large amounts of RAM (random access memory) and other system resources. This limits the numbers of Vusers that can be run on any load generator.

Alternatively, if you run each Vuser as a **thread**, the Controller launches only one instance of the driver program (such as *mdrv.exe*), for every 50 Vusers (by default). This driver process/program launches several Vusers, each Vuser running as a thread. These threaded Vusers share segments of the memory of the parent driver process. This eliminates the need for multiple re-loading of the driver program/process saves much memory space, thereby enabling more Vusers to be run on a single load generator.

To configure these options, open the runtime settings (F4) and select the **General > Miscellaneous** node.

Runtime Settings

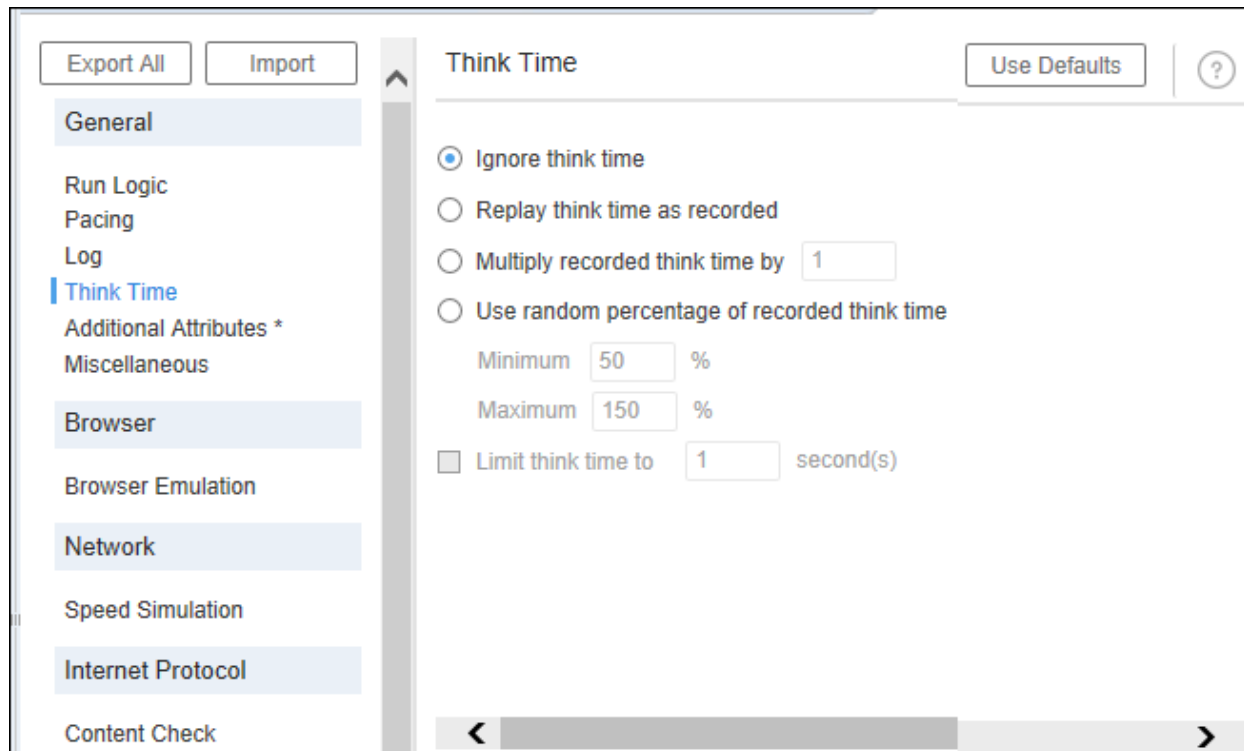
This section contains a variety of topics relating to runtime settings (**Replay > Runtime Settings**).

Runtime Settings Overview

Runtime settings define the way a Vuser script runs, and can be configured before or after you record a script. These settings are stored in files located in the Vuser script folder. Runtime settings are applied to Vusers when you run a script using VuGen, the Controller, Performance Center, or Business Process Monitor.

Configuring runtime settings allows you to emulate different kinds of user activity. For example, you can emulate a user who responds immediately to output from the server, or a user who stops and thinks before each response. You can also configure the runtime settings to specify how many times the Vuser should repeat its set of actions.

A blue line adjacent to the runtime setting, indicates the current view. The following example indicates that the **Think Time** runtime setting is active.



You can export runtime settings to a JSON file and import them into another script, instead of having to set them repeatedly for each script. You can also revert the runtime settings back to the default values. For more information, see ["Import and Export Runtime Settings" on page 304](#).



Different combinations of runtime settings are available for each protocol. When you open the runtime settings, the relevant runtime setting views for that protocol are listed.

Runtime settings are now a script entity. This allows you to copy or export them for use with other scripts using the shortcut menu.



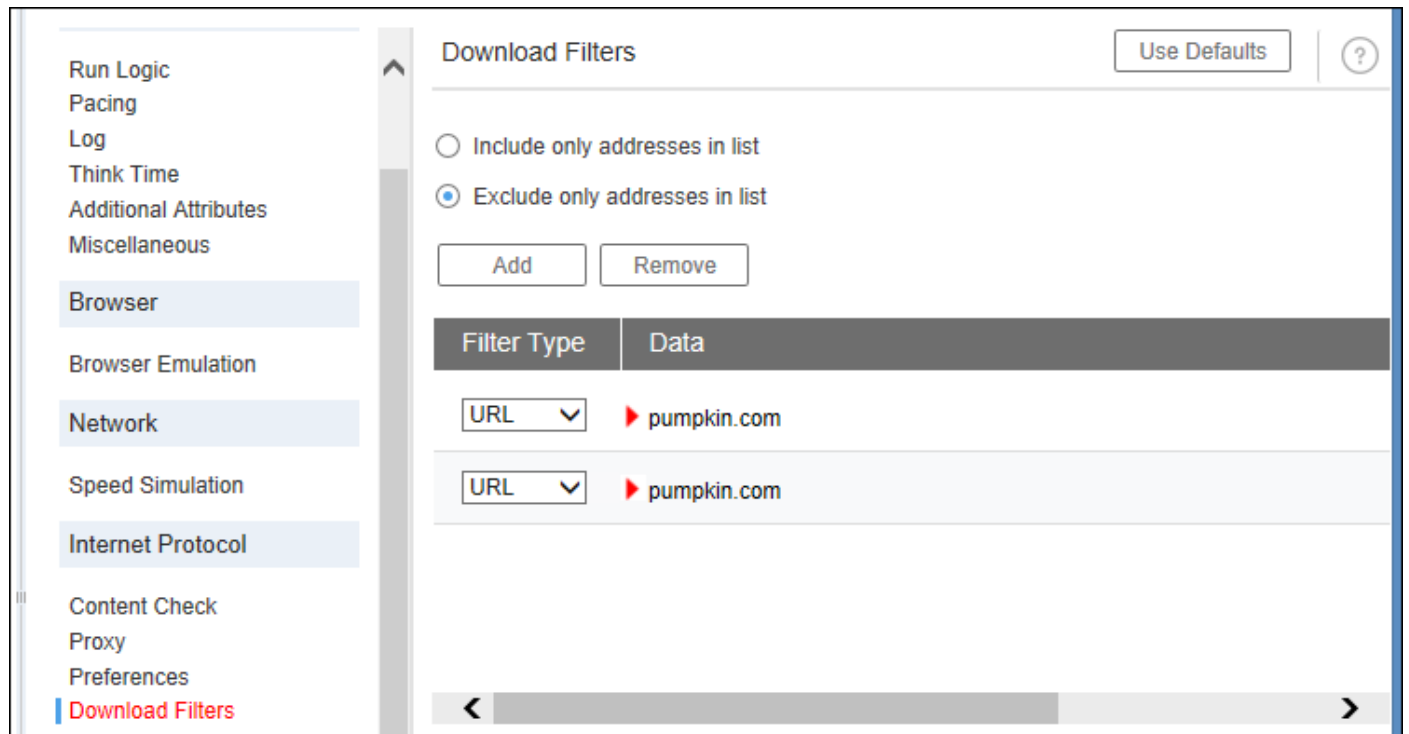
Tip: Descriptions of the individual runtime settings are available within the runtime settings views, by hovering the mouse over a runtime setting field name.

Runtime Setting Value Validation

The Solution Explorer indicates that one or more runtime settings have illegal values, with a warning icon  in place of the standard Runtime Setting node icon, .

If you enter a value above the maximum allowed value, VuGen automatically substitutes it with the maximum allowed value. If you enter a value below the minimum allowed value, VuGen automatically substitutes it with the minimum allowed value.

A red highlighted value, indicates that one of the values in the view is invalid. The following example shows duplicate filter names in the **Download Filters** view.



See also:

- ["Runtime Settings Views" below](#)

Runtime Settings Views

This page describes the runtime settings views.

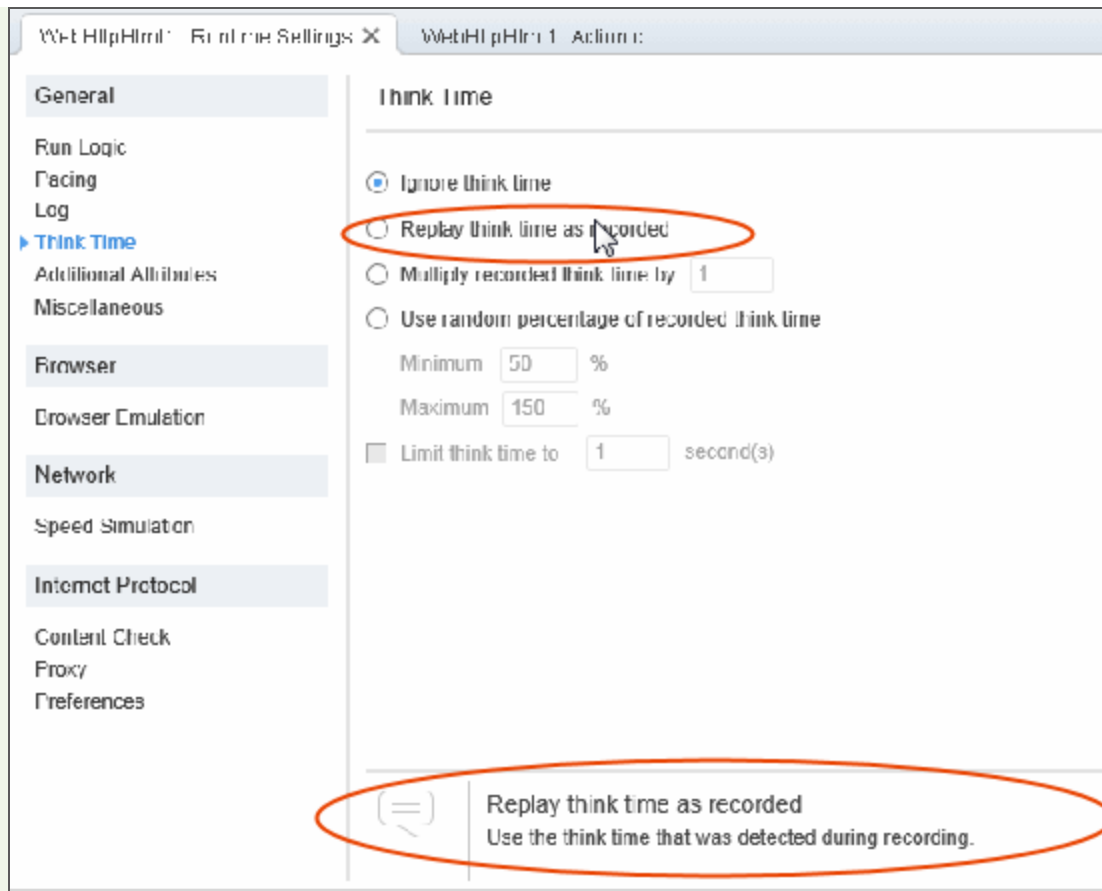
Access Runtime Settings View

To open the runtime settings do one of the following:

- In the Solution Explorer, double-click on the name of a **Runtime Settings** view.
- Select the menu item **Replay > Runtime Settings**.
- Press the shortcut key, F4.

Note:

- Internet Explorer 10 or higher must be installed in order to display the runtime settings.
- For an overview of the runtime settings, see ["Runtime Settings Overview" on page 283](#). Descriptions of the individual runtime settings are available within the runtime settings views, by hovering the mouse over a runtime setting field name.



For some runtime settings, there is additional information on the settings included in ["Runtime Settings View Descriptions"](#) below.

Runtime Settings View Descriptions

View	Description
.NET Environment View	Enables you to set the runtime settings for .NET Vuser scripts.
Additional Attributes View	Enables you to provide additional arguments for a Vuser script. The Additional Attributes settings apply to all Vuser script types. You specify command line arguments that you can retrieve at a later point during the test run, using Command Line Parsing functions. Using this view, you can pass external parameters to prepared scripts.
Browser Emulation View	Enables you to configure the browser related runtime settings.

View	Description
Browser View (TruClient - Web)	<p>Enables you to configure settings for the TruClient browsers, for scripts that you run in load mode.</p> <p>Additional information:</p> <ul style="list-style-type: none"> Settings that you modify in this view only affect TruClient Vusers in load mode. These settings correspond to those in the Browser Settings tab in the TruClient General Settings Dialog Box. However, the settings that you modify in the TruClient General Settings Dialog Box only affect interactive mode. For details on this dialog box, see the TruClient Help Center (select the relevant version). When you save your script in interactive mode, the settings that you modified in the Browser Settings tab are applied to these Load Runtime settings.
Citrix Configuration View	Enables you to set the Citrix configuration runtime settings.
Citrix Synchronization View	Enables you to set the Citrix synchronization runtime settings.
Client Emulation View	Enables you to set the Oracle NCA runtime settings.
Content Check View	<p>Enables you to check websites for content during runtime. You can set Content Check runtime settings for Web - HTTP/HTML and other Internet protocols.</p> <p>Additional information:</p> <p>You use the Content Check settings to check the contents of a page for a specific string. This is useful for detecting non-standard errors. In normal operations, when your application server fails, the browser displays a generic HTTP error page indicating the nature of the error. The standard error pages are recognized by VuGen and treated as errors, causing the script to fail. Some application servers, however, issue their own error pages that are not detected by VuGen as error pages. The page is sent by the server and it contains a formatted text string, stating that an error occurred.</p> <p>For example, suppose that your application issues a custom page when an error occurs, containing the text ASP Error. You instruct VuGen to look for this text on all returned pages. When VuGen detects this string, it fails the replay.</p> <p>Note: VuGen searches the body of the pages—not the headers.</p>

View	Description
DFE Chain Configuration View	Enables Data Format Extensions during replay.
Download Filters View	Enables you to set the download filters for a script.
Flex Configuration View	Enables you to set an external JVM (Java Virtual Machine) path and other runtime settings.
Flex Externalizable View	Enables you to configure runtime setting for externalizable objects in Flex scripts.
Flex RTMP View	Enables you to set the Flex RTMP runtime settings.
Java Classpath View	Enables you to specify the location of additional classes that were not included in the system's classpath environment variable. You may need these classes to run Java applications and ensure proper replay.
Java VM View	Enables you to set the Java VM runtime settings.
JMS Advanced View	Enables you to set the JMS advanced runtime settings.
Log View	Enables you to configure the amount and types of information that are recorded in the log.
Log View (TruClient)	Enables you to configure the amount and types of information that are reported to a log for TruClient scripts.
Miscellaneous View	<p>Enables you to set miscellaneous runtime settings.</p> <p>Automatic transactions are not relevant for HPE Application Performance Management.</p> <p>Tips:</p> <ul style="list-style-type: none"> • It is not recommended to enable both the Continue on Error and Generate Snapshot on Error options in a load test environment. This configuration may adversely affect the Vusers' performance. • If you require the Vusers to generate breakdown data for diagnostics (J2EE) during the test run, do not use automatic transactions. Instead, manually define the beginning and end of each transaction. • For considerations on whether to run a Vuser as a process or thread, see "Running a Vuser as a Process or Thread" on page 283.
Mobile Device View	Enables you to select mobile device properties when recording a TruClient - Mobile Web script.

View	Description
MQTT Configuration View	Enables you to configure various MQTT runtime settings, some of which are used as the default values for arguments in the script.
Other Settings View (TruClient)	Enables you to configure snapshot generation and action on error for TruClient.
Pacing View	Enables you to control the time between iterations. The pace tells the Vuser how long to wait between iterations of your actions.
Preferences View	Enables you to set various Internet-related runtime settings. For information about the Internet Preferences runtime settings, see " Preferences View - Internet Protocol " on page 291.
Proxy View	Enables you to set the proxy server connection settings. If you select the option to use the default HTTP proxy settings: <ul style="list-style-type: none"> When running on Windows, the Internet Explorer proxy settings are used. To use this option, your default browser cannot be Firefox. If it is, uninstall Firefox and then select another default browser. When running on Linux, the proxy values in the HTTP_PROXY or HTTPS_PROXY environment variables are used. You must manually create these environment variables on the Linux computer where your script runs.
RDP Advanced View	Enables you to set the RDP advanced runtime settings. Tip: Disable the options that are not essential for your test, in order to conserve system resources on the remote desktop server.
RDP Agent View	Enables you to set the RDP Agent runtime settings. Tip: For the RDP agent log folder option—if no folder is specified and the agent log destination was set to File , the log is saved in the temp folder of the user on the server.
RDP Configuration View	Enables you to set the RDP configuration runtime settings.
RDP Synchronization View	Enables you to set the RDP synchronization runtime settings.
Replay View (TruClient)	Enables you to set the runtime settings for replay of TruClient scripts.

View	Description
RTE View	<p>Enables you to set the RTE runtime settings.</p> <p>You can use the TE_typing_style function to override the Delay settings for a portion of a Vuser script.</p> <p>Tip: In the Delay before typing option, the delay settings determine how Vusers execute TE_type functions.</p> <ul style="list-style-type: none"> • First key. Specifies the time in milliseconds, that a Vuser waits before entering the first character of a string. • Subsequent keys. Specifies the time in milliseconds, that a Vuser waits between submitting successive characters.
Run Logic View	Enables you to set the run logic runtime settings.
SAP GUI > General View	<p>Enables you to set the SAP GUI runtime settings.</p> <p>Performance settings:</p> <ul style="list-style-type: none"> • Show SAP Client during replay. This option shows an animation of the actions in the SAP client during replay. The benefit of this, is that you can closely follow the actions of the Vuser and see how the forms are filled out. This option, however, requires additional resources and may affect the performance of your load test. • Create snapshots during replay. Captures both ActiveScreen snapshots and regular snapshots while a script runs. <p>ActiveScreen snapshots contain control ID information for all active objects. ActiveScreen snapshots differ from regular snapshots in that ActiveScreen snapshots allow you to see which objects were recognized by VuGen in the SAP GUI client. As you move your mouse across an ActiveScreen snapshot, VuGen highlights the detected objects. You can then add new steps to the script directly from within the snapshot. ActiveScreen snapshots also enable you to add steps interactively from within the snapshot for a specific object. For more information, see "Enhance SAP GUI Scripts" on page 650.</p> <p>Note: Disabling replay snapshots may improve the script replay speed and save storage space.</p>
Server (TruClient Native Mobile)	Enables you to specify the server from which you want to collect data and the credentials.

View	Description
Shared DLLs View	Enables you to modify the list of shared DLLs after you record a Vuser script. If a DLL is included in the list of shared DLLs, when the Vuser script is run and requires a particular DLL, the Vuser will access the DLL in its shared location – the DLL will not be copied to the load generator. Adding a DLL to the list of shared DLLs therefore saves hard-drive space on the load generator when a Vuser is run.
Silverlight Services View	Enables you to view the WSDL files associated with your script and modify their settings for the replay phase.
Speed Simulation View	Enables you to configure bandwidth runtime settings.
Speed Simulation View (TruClient)	Enables you to configure bandwidth for the TruClient Web and Mobile Web protocols.
Think Time View	Enables you to configure the think time settings, controlling the time that a VuGen waits between actions. These settings are designed to help you emulate a real user.

Preferences View - Internet Protocol

The Preferences view runtime settings (**Replay > Runtime Settings > Internet Protocol > Preferences**) enable you to set various Internet-related options.

This view is available only for specific protocols. When you open the runtime settings, only the relevant views are displayed.


For general information about runtime settings, see ["Runtime Settings Overview" on page 283](#).

The user interface elements are described below:

Checks

UI Element	Description
Enable image and text checks	Allows the Vuser to perform verification checks during replay by executing the verification functions web_find or web_image_check . This option only applies to statements recorded in HTML-based mode. Vusers running with verification checks use more memory than Vusers who do not perform checks. Default value: Disabled

Web Performance Graph Generation

UI Element	Description
Hits per Second	Instructs a Vuser to collect data for Web Performance graphs.
Pages per Second	Select the types of graph data for the Vuser to collect to view the (Throughput) graphs during test execution using the online monitors and after test execution using the Analysis. You view the Component Breakdown graph after test execution using the Analysis.
Response Bytes per Second	<div>  Note: If you do not use the Web performance graphs, disable these options to conserve memory. </div>

Advanced

UI Element	Description
Use WinInet replay instead of Sockets (Windows only)	<p>Instructs VuGen to use the WinInet replay engine instead of the standard Sockets replay. VuGen has two HTTP replay engines: Sockets-based (default) or WinInet based. The WinInet is the engine used by Internet Explorer and it supports all of the features incorporated into the IE browser. The limitations of the WinInet replay engine are that it is not scalable and does not support Linux. In addition, when working with threads, the WinInet engine does not accurately emulate the modem speed and number of connections. VuGen's proprietary sockets-based replay is a lighter engine that is scalable for load testing. It is also accurate when working with threads. The limitation of the sockets-based engine is that it does not support SOCKS proxy. If you are recording in that type of environment, use the WinInet replay engine.</p> <p>Default value: disabled (socket-based replay engine).</p>
Include File name and line in automatic transaction names.	Creates unique transaction names for automatic transactions by adding file name and line number to the transaction name.

UI Element	Description
List non-critical resource errors as warnings	<p>Returns warnings for actions that fail during test replay, when the actions are performed on non-critical resources, thereby enabling the test to replay successfully.</p> <p>For details on how a resource is classified as either critical or non-critical, see "Defining Non-Critical Resources" on page 303.</p> <p>If you want the failure of non-critical resources to be errors and thereby fail your test, you can disable this option. This option is enabled by default.</p> <p>You can set a content-type to be critical by adding it to the list of Non-Resources. For more information, see "Non-Resources Dialog Box" on page 181.</p>
Save snapshot resources locally	Saves the snapshot resources to files on the local machine.
Enable Dynatrace monitoring	<p>Enables LoadRunner to push data to the Dynatrace monitor on the AUT server during the scenario run.</p> <p>When enabled, the x-dynaTrace header required by Dynatrace is added to each HTTP request.</p> <p>Note: When a script runs on a Linux load generator, the transaction name is not displayed on the Dynatrace monitor.</p>

HTTP

UI Element	Description
HTTP version	<p>Specifies which version HTTP to use: version 1.0 or 1.1. This information is included in the HTTP request header whenever a Vuser sends a request to a Web server.</p> <p>HTTP 1.1 supports the following features:</p> <ul style="list-style-type: none"> • Persistent Connections—see "Keep-Alive HTTP connections" below. • HTML compression—see Accept Server-Side Compression below. • Virtual Hosting—multiple domain names sharing the same IP address.

UI Element	Description
Keep-Alive HTTP connections	<p>Keep-alive is a term used for an HTTP extension that allows persistent or continuous connections. These long-lived HTTP sessions allow multiple requests to be sent over the same TCP connection. This improves the performance of the Web server and clients.</p> <p>The keep-alive option works only with Web servers that support keep-alive connections. This setting specifies that all Vusers that run the Vuser script have keep-alive HTTP connections enabled.</p> <p>Default value: Enabled</p>
Include Accept-Language request header	<p>Provides a comma-separated list of accepted languages. For example, en-us, fr, and so forth. For more details, see "Page Request Header Language" on page 862.</p>
Mark HTTP errors as warnings	<p>Issues a warning instead of an error upon failing to download resources due to an HTTP error.</p>
HTTP-request connect timeout (sec)	<p>The time, in seconds, that a Vuser will wait for the connection of a specific HTTP request within a step before aborting. Timeouts provide an opportunity for the server to stabilize and respond to the user.</p> <p>Maximum value: 32000 seconds</p>
HTTP-request receive timeout (sec)	<p>The time, in seconds, that a Vuser will wait to receive the response of a specific HTTP request within a step before aborting. Timeouts provide an opportunity for the server to stabilize and respond to the user.</p> <p>Maximum value: 32000 seconds</p>
HTTP Keep-Alive timeout (sec)	<p>A time limit within which some activity must be performed on an HTTP connection. If this timeout is reached, the connections is closed during replay.</p>
Request zlib headers	<p>Sends request data to the server with the zlib compression library headers. By default, requests sent to the server include the zlib headers. This option lets you emulate non-browser applications that do not include zlib headers in their requests.</p> <p>Default value: Enabled</p>

UI Element	Description
Accept server-side compression	<p>Indicate to the server that the replay can accept compressed data. The available options are: None (no compression), gzip (accept gzip compression), gzip, deflate (accept gzip or deflate compression), and deflate (accept deflate compression). Note that by accepting compressed data, you may significantly increase the CPU consumption.</p> <p>Default value: Accept gzip and deflate compression.</p> <p>To manually add compression, enter the following function at the beginning of the script:</p> <pre>web_add_auto_header("Accept-Encoding", "gzip");</pre> <p>To verify that the server sent compressed data, search for the string Content - Encoding: gzip in the section of the server's responses of the replay log. The log also shows the data size before and after decompression.</p>
Delete unreferenced cache entries	<p>Delete cache entries that have not been referenced within the specified number of iterations. Set to zero (0) to never delete cache entries.</p>

General

UI Element	Description
Create snapshots during replay	<p>Create snapshots during replay.</p> <div> <p>Note: Disabling replay snapshots will improve the replay speed. However, snapshot-dependent features such as DFE and correlations, will not be able to use data captured during the replay. This may cause unstable behavior.</p> </div>
DNS caching	<p>Instructs the Vuser to save a host's IP addresses to a cache after resolving its value from the Domain Name Server. This saves time in subsequent calls to the same server. In situations where the IP address changes, as with certain load balancing techniques, be sure to disable this option to prevent Vuser from using the value in the cache.</p> <p>Default value: Enabled</p>
Convert to/from UTF-8	<p>Converts received HTML pages and submitted data from and to UTF-8. You enable UTF-8 support in the recording options. For more information, see "Recording Options" on page 141.</p> <p>Default value: No</p>

UI Element	Description
Charset to use for converting HTML	The character set to use to convert received HTMLs and submitted data from/to the set charset. This option is ignored if you enabled the previous option, '\Convert to/from UTF-8\'.
Mark step timeouts caused by resources as a warning	Issues a warning instead of an error when a timeout occurs due to a resource that did not load within the timeout interval. For non-resources, VuGen issues an error. Default value: Disabled
Parse HTML content-type	When expecting HTML, parse the response only when it is the specified content-type: HTML , text/html , TEXT any text, or ANY , any content-type. Note that text/xml is not parsed as HTML. Default value: TEXT
Step download timeout (sec)	The time that the Vuser will wait before aborting a step in the script. This option can be used to emulate a user behavior of not waiting for more than x seconds for a page. Maximum value: 32000 seconds The timeout settings are primarily for advanced users who have determined that acceptable timeout values should be different for their environment. The default settings should be sufficient in most cases. If the server does not respond in a reasonable amount of time, check for other connection-related issues, rather than setting a very long timeout which could cause the scripts to wait unnecessarily.
Network buffer size	Sets the maximum size of the buffer used to receive the HTTP response. If the size of the data is larger than the specified size, the server will send the data in chunks, increasing the overhead of the system. When running multiple Vusers from the Controller, every Vuser uses its own network buffer. This setting is primarily for advanced users who have determined that the network buffer size may affect their script's performance. The default is 12K bytes. The maximum size is 0x7FFF FFFF.
Print NTLM information	Print information about the NTLM handshake to the standard log.
Print SSL information	Print information about the SSL handshake to the standard log.
SSL version	The version of SSL used by your application.

UI Element	Description
Maximum number of failure-matches to list as errors	<p>Limit the number of content-check failures that are issued as errors, where a failure is indicated by the appearance of a string (Fail=Found). This applies to match criteria using a left and right boundary. All subsequent matches are listed as informational messages.</p> <p>Default value: 10 matches</p>
Maximum redirection depth	<p>The maximum number of allowed redirections.</p> <p>Default value: 10</p>
Maximum number of 'META Refresh' on a single page	<p>The maximum number of times that a META refresh can be performed per page.</p> <p>Default value: 2</p>
Consider ContentCheck values as UTF-8 encoded	<p>Store the values in the ContentCheck XML file in UTF-8.</p> <p>Default value: Disabled</p>
Limit the Tree view request body to	<p>Limit the number of request body bytes displayed in Tree-View. Set to zero (0) for no limit.</p>
Limit the stored snapshot to	<p>Limit the size of each snapshot file to a specific number of kilobytes. Enter 0 to indicate no limit.</p>
IP version	<p>The IP version to be used: IPv4, IPv6 or automatic selection. The default value is IPv4.</p>
web_sync retry interval	<p>The time to wait (in milliseconds) between testing the condition that yields false and the next retry.</p> <p>Default value: 1000</p>
web_sync retry timeout	<p>The maximum time (in milliseconds) during which retries are allowed. If the computed timeout exceeds the step timeout (as determined by the 'Step download timeout' setting), the latter is used.</p>

UI Element	Description
WebSocket callback interval	The time interval in milliseconds, before repeating a call to a WebSocket callback handler. This must be a non-zero value.
Prefetch and prerender callback timer interval	The time interval in milliseconds, before repeating a call to Prefetch and Prerender callback handlers. This must be a non-zero value.

Authentication

UI Element	Description
Add a fixed delay upon authentication	<p>Automatically adds think time to the Vuser script for emulating a user entering authentication information (username and password). This think time will be included in the transaction time.</p> <p>Default value: 0</p>
Disable NTLM2 session security	<p>Use full NTLM 2 handshake security instead of the more basic NTLM 2 session security response.</p> <p>Default value: No</p>
Use the native Windows NTLM implementation	<p>Use the Microsoft Security API for NTLM authentication instead of the indigenous one.</p> <p>Default value: No</p>
Override credentials in a Windows native NTML implementation	Use the credentials provided by the user at logon.
Enable integrated authentication	<p>Enable Kerberos-based authentication. When the server proposes authentication schemes, use Negotiate in preference to other schemes.</p> <p>Default value: No</p>
Induce heavy KDC load	<p>Do not reuse credentials obtained in previous iterations. Enabling this setting will increase the load on the KDC (Key Distribution Server). To lower the load on the server, set this option to Yes in order to reuse the credentials obtained in previous iterations. This option is only relevant when Kerberos authentication is used.</p> <p>Default value: No</p>

UI Element	Description
Use canonical name in SPN	Use the canonical name instead of the original hostname retrieved from the URL, to generate SPN (Service Principal Name). Default value: Yes
Append non-default port to SPN	Append the port number to the SPN, if the specified port is a non-standard one (neither 80 nor 443). Default value: No
Enable retrieving keys from nCipher HSM	Enables Vusers to retrieve private keys from the nCipher HSM (Hardware Security Module). This option loads and initializes the CHIL engine necessary to retrieve these keys. Default value: Yes

Logging

UI Element	Description
Print buffer line length	Line length for printing request/response header/body and/or JavaScript source, disabling wrapping.
Print buffer escape for binary zeros only	<ul style="list-style-type: none"> • Yes. Escape only binary zeros when printing request/response headers/body and/or JavaScript source. • No. Escape any unprintable/control characters.
Limit the maximum response size written to the log	Limits the size of the log containing the response data.

JavaScript

UI Element	Description
Enable JavaScript debugging mode	Only visible for Web-based Vuser Scripts generated in the JavaScript language. Enables debugging for the replay of Vuser scripts. This only applies to the replay in VuGen—not the Controller. Enabling this option may impact replay performance.
Enable running JavaScript code	Only visible for Vuser Scripts generated in the C language. Enables the replay of Web JavaScript steps, such as web_js_run() and web_js_reset() . This option creates a JavaScript runtime engine even in the there are no JavaScript steps in the script.

UI Element	Description
JavaScript Engine runtime size	Only visible for Vuser Scripts generated in the C language. The memory size in kilobytes, to allocate for the JavaScript engine runtime. One runtime engine will be created for all Vusers in a process. Default: 51200 KB
JavaScript Engine stack size per thread	Only visible for Vuser Scripts generated in the C language. The memory size in kilobytes, to allocate for each Vuser thread in the JavaScript engine. Default: 32 KB

Click & Script Preferences

UI Element	Description
General	<ul style="list-style-type: none"> • Home Page URL. The URL of the home page that opens with your browser (default is about:blank). • DOM-based snapshots. Instructs VuGen to generate snapshots from the DOM instead of from the server responses. Default value: Yes • Charset conversions by HTTP. Perform charset conversions by the 'Content-Type:....; charset=...' HTTP response header. Overrides 'Convert from /to UTF-8.' • Reparse when META changes charset. Reparse HTML when a META tag changes the charset. Effective only when Charset conversions by HTTP is enabled. Auto means reparsing is enabled only if it used in the first iteration. • Fail on JavaScript error. Fails the Vuser when a JavaScript evaluation error occurs. Default value: No (issue a warning message only after a JavaScript error, but continue to run the script). • Initialize standard classes for each new window project. When enabled, the script—the src compiled script, will not be cached. • Ignore acted on element being disabled. Ignore the element acted on by a Vuser script function being disabled.

UI Element	Description
Timers	<ul style="list-style-type: none"> • Optimize timers at end of step. When possible, executes a setTimeout/setInterval/<META refresh> that expires at the end of the step before the expiration time. Default value: Yes • Single setTimeout/setInterval threshold (seconds). Specifies an upper timeout for the window.setTimeout and window.setInterval methods. If the delay exceeds this timeout, these methods will not invoke the functions that are passed to them. This emulates a user waiting a specified time before clicking on the next element. Default value: 5 seconds • Accumulative setTimeout/setInterval threshold (seconds). Specifies a timeout for the window.setTimeout and window.setInterval methods. If the delay exceeds this timeout, additional calls to window.setTimeout and window.setInterval will be ignored. The timeout is accumulative per step. Default value: 30 seconds • Reestablish setInterval at end of step. 0 = No; 1 = Once; 2 = Yes. • Limit no-network timers at end of step: Limit the number of setTimeout/setInterval specified script evaluations at the end of a step when no network requests are issued. Set to zero (0) for no limit. The default value is 100. This limit is only used when 'Optimize timers at end of step' is enabled.
History	<ul style="list-style-type: none"> • History support. Enables support for the window.history object for the test run. The options are Enabled, Disabled, and Auto. The Auto option instructs Vusers to support the window.history object only if it was used in the first iteration. Note that by disabling this option, you improve performance. Default value: Auto • Maximum history size. The maximum number of steps to keep in the history list. Default value: 100 steps

UI Element	Description
Navigator Properties	<ul style="list-style-type: none"> • navigator.browserLanguage. The browser language set in the navigator DOM object's browserLanguage property. Default value: The recorded value. Scripts created with older recording engines use en-us by default. • navigator.systemLanguage. The system language set in the navigator DOM object's systemLanguage property. Default value: The recorded value. Scripts created with older recording engines use en-us by default. • navigator.userLanguage. The user language set in the navigator DOM object's userLanguage property. Default value: The recorded value. Scripts created with older recording engines use en-us by default.
Screen Properties	<ul style="list-style-type: none"> • screen.width Sets the width property of the screen DOM object in pixels. Default value: 1024 pixels • screen.height Sets the height property of the screen DOM object in pixels. Default value: 768 pixels • screen.availWidth Sets the availWidth property of the screen DOM object in pixels. Default value: 1024 pixels • screen.availHeight. Sets the availHeight property of the screen DOM object in pixels. Default value: 768 pixels

UI Element	Description
Memory Management	<ul style="list-style-type: none"> • Default block size for DOM memory allocations. Sets the default block size for DOM memory allocations. If the value is too small, it may result in extra calls to malloc, slowing the execution times. Too large a block size, may result in an unnecessarily big footprint. Default value: 16384 bytes • Memory Manager for dynamically-created DOM objects.Yes—Use the Memory Manager for dynamically-created DOM objects. No—Do not use the Memory Manager, for example when multiple DOM objects are dynamically created in the same document as under SAP. Auto—Use the protocol recommended (default Yes for all protocols except for SAP). • JavaScript Runtime memory size (KB). Specifies the size of the JavaScript runtime memory in kilobytes. Default value: 256 KB • JavaScript Stack memory size (KB). Specifies the size of the JavaScript stack memory in kilobytes. Default value: 32 KB
Web Javascript	<ul style="list-style-type: none"> • Enable running Javascript code. Yes—Enables running web Javascript steps, such as <code>web_js_run()</code> and <code>web_js_reset()</code>. No—Web Javascript steps cannot be run. Note that enabling this option causes the creation of a Javascript Engine Runtime, even if there are no Javascript steps in the script. Default value: No • Javascript Engine runtime size (KB). Specifies the size of the Javascript Engine Runtime memory in kilobytes. One Runtime will be created for all Vusers in a process. Default value: 10240 KB • Javascript Engine stack size per-thread (KB). Specifies the size of each Vuser thread in the Javascript Engine memory, in kilobytes. Default value: 32 KB

Defining Non-Critical Resources

You can define resources used in your test to be non-critical, so that the test can replay successfully even when actions performed on these resources fail. For example, an image or a Java applet that failed to download may not be critical to your business process.

Prevent unnecessary failures due to errors with non-critical resources

- In the Runtime Settings Preferences view, the field [List non-critical resource errors as warnings](#) is enabled (default value). See "[Preferences View - Internet Protocol](#)" on page 291.

- You classify a resource as non-critical, by defining the value of the Resource attribute in the function, as described below.

Resource classification

The classification of a resource as critical or non-critical depends on the value of the **Resource** attribute in the function, as described below.

1. If **Resource** is specified:
 - Resource = 0: The resource is always critical.
 - Resource = 1: The resource is always non-critical.
2. If **Resource** is not specified:
 - If Method is not Get: The resource is always critical.
 - If Method is Get:

RecContentType is omitted	RecContentType is text/html	RecContentType is not text/html
Critical	Critical	Not critical

Example

The resource in the following function is non-critical because Resource is 1:

```
web_url("webcode.exe",  
        "URL=https://Example/asp/monitora.asp",  
        "Resource=1",  
        "RecContentType=gif",  
        "Referer=",  
        "Snapshot=t1.inf",  
        LAST);
```

Import and Export Runtime Settings

This topic describes how to import and export runtime settings. This functionality is available for all Vuser protocols.

Export runtime settings

1. Open a script and double-click **Runtime Settings** in the Solution Explorer.
2. In the Runtime Settings view, click **Export All**.
3. In the Save As dialog box, choose a location for the JSON file that will store the runtime settings and click **Save**.

Import runtime settings

1. Open a script and double-click **Runtime Settings** in the Solution Explorer.
2. In the Runtime Settings view, click **Import**.
3. In the Open dialog box, select a JSON file containing the runtime settings that you exported earlier and click **Open**.
4. Save the script.

Revert runtime settings to the default settings

1. In the Solution Explorer pane, select the runtime settings node of the script to be changed and navigate to the required setting.
2. Click the **Use Defaults** button.
3. Save the script. Only the defaults for the displayed node are changed. If you want to revert to the default settings for all the runtime settings, you must repeat the above steps for each runtime setting node.



See also:

- ["Runtime Settings Overview" on page 283](#)

Configure Runtime Settings Manually

To configure Vuser runtime settings, you modify the *default.cfg* and *default.usp* files created with the script. These runtime settings correspond to VuGen's runtime settings. (See ["Runtime Settings Overview" on page 283](#).) The *default.cfg* file contains the setting for the General, Think Time, and Log options. The *default.usp* file contains the setting for the Run Logic and Pacing.

General Options

There is one General option for Linux Vuser scripts:

ContinueOnError instructs the Vuser to continue when an error occurs. To activate the option, specify 1. To disable the option, specify 0.

In the following example, the Vuser will continue on an error.

```
[General]
ContinueOnError=1
```

Think Time Options

You can set the think time options to control how the Vuser uses think time during script execution. You set the parameters Options, Factor, LimitFlag, and Limit parameters according to the following chart.

Option	Options	Factor	LimitFlag	Limit
Ignore think time	NOTHINK	N/A	N/A	N/A
Use recorded think time	RECORDED	1.000	N/A	N/A
Multiply the recorded think time by...	MULTIPLY	number	N/A	N/A
Use random percentage of recorded think time	RANDOM	range	lowest percentage	upper percentage
Limit the recorded think time to...	RECORDED/ MULTIPLY	number (for MULTIPLY)	1	value in seconds

To limit the think time used during execution, set the `LimitFlag` variable to 1 and specify the think time `Limit`, in seconds.

In the following example, the settings tell the Vuser to multiply the recorded think time by a random percentage, ranging from 50 to 150.

```
[ThinkTime]
Options=RANDOM
Factor=1
LimitFlag=0
Limit=0
ThinkTimeRandomLow=50
ThinkTimeRandomHigh=150
```

Log Options

You can set the log options to create a brief or detailed log file for the script's execution.

```
[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

You set the parameters `LogOptions`, `MsgClassData`, `MsgClassParameters`, and `MsgClassFull` variables according to the following chart:

Logging Type	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
Disable Logging	LogDisabled	N/A	N/A	N/A
Standard Log	LogBrief	N/A	N/A	N/A

Parameter Substitution (only)	LogExtended	0	1	0
Data Returned by Server (only)	LogExtended	1	0	0
Advanced Trace (only)	LogExtended	0	0	1
All	LogExtended	1	1	1

In the following example, the settings tell the Vuser to log all data returned by the server and the parameters used for substitution.

```
[Log]
LogOptions=LogExtended
MsgClassData=1
MsgClassParameters=1
MsgClassFull=0
```

Iterations and Run Logic

You can set the Iteration options to perform multiple iterations and control the pacing between the iterations. You can also manually set the order of the actions and their weight. To modify the run logic and iteration properties of a script, you must edit the *default.usp* file.

To instruct the Vuser to perform multiple iterations of the Actions section, set `RunLogicNumOfIterations` to the appropriate value.

To control the pacing between the iterations, set the `RunLogicPaceType` variable and its related values, according to the following chart:

Pacing	RunLogicPaceType	Related Variables
As soon as possible	ASAP	N/A
Wait between Iterations for a set time	Const	RunLogicPaceConstTime
Wait between iterations a random time	Random	RunLogicRandomPaceMin, RunLogicRandomPaceMax
Wait after each iteration a set time	ConstAfter	RunLogicPaceConstAfterTime
Wait after each iteration a random time	After	RunLogicAfterPaceMin, RunLogicAfterPaceMax

In the following example, the settings tell the Vuser to perform four iterations, while waiting a random number of seconds between iterations. The range of the random number is from 60 to 90 seconds.

```
[RunLogicRunRoot]
MercIniTreeFather=""
MercIniTreeSectionName="RunLogicRunRoot"
RunLogicRunMode="Random"
RunLogicActionOrder="Action,Action2,Action3"
RunLogicPaceType="Random"
RunLogicRandomPaceMax="90.000"
RunLogicPaceConstTime="40.000"
RunLogicObjectKind="Group"
RunLogicAfterPaceMin="50.000"
Name="Run"
RunLogicNumOfIterations="4"
RunLogicActionType="VuserRun"
RunLogicAfterPaceMax="70.000"
RunLogicRandomPaceMin="60.000"
MercIniTreeSons="Action,Action2,Action3"
RunLogicPaceConstAfterTime="30.000"
```

Bookmarks Overview

When you edit a Vuser script, you can use bookmarks to navigate between sections of the script. When you add a bookmark, a bookmark icon is added to the left of the selected line in your script.

The Bookmarks pane displays a list of all bookmarks that exist in the Vuser script. Using the Bookmarks pane, you can:

- Navigate to the location of the bookmark in your script.
- Navigate between consecutive bookmarks in the pane.
- Delete an individual bookmark.
- Clear all bookmarks.



See also:

- ["Use Bookmarks" on the next page](#)

Run a Vuser Script from a Command Prompt

This task describes how to replay a Vuser script on a machine from a command prompt or from the Windows Run dialog box—without the VuGen user interface.

To send command line parameters to a Vuser from within VuGen, add the attributes and their values in the Runtime Settings dialog box. For details, see the **General > Additional Attributes** view.

When you run a script from the command line, VuGen replays it in its basic form. The command line replay will not capture anything related to the UI, since it is absent in this type of run. This excludes the capturing of snapshots and similar runtime settings.

To run a script from a command line or the Run dialog box:

1. Open a **Command Prompt** window, or select **Start > Run** to open the Run dialog box.
2. Type **mdrv** followed by the script name, using the following syntax:

```
<installation_dir>/bin/mdrv.exe -usr <script_name>
```

where **script_name** is the full path to the *.usr* script file, for example, **c:\temp\mytest\mytest usr.**

3. Add other command line options and arguments.
4. Click **Enter**. The **mdrv** program runs a single instance of the script without the user interface. The output files provide the runtime information.

For a complete list of the command line options, type **mdrv** at a command prompt from VuGen's **bin** folder, without any arguments.

Note: The Linux command line utility, *run_db_vuser*, does not yet support many of the standard Windows command line options. For details, see ["Run a Vuser Script from a Linux Command Line" on page 849](#).

The following examples provide common usages of a command line expression:

- You can specify the load generator, as well as indicate the number of times to run the script as indicated by the following example:

```
script1 -host pc4 -loop 5
```

- Specify a location for the output files. For example:

```
-out c:\tmp\vuser
```

- Specify arguments to pass to your script by using the following format:

```
script_name -arg_name arg_value -arg_name arg_value
```

You can retrieve the command line values by parsing the command line during replay, using the parsing functions, such as **lr_get_attrib_double**. For details, see the [Function Reference \(Help > Function Reference\)](#).




Tip: To further customize your run, set the runtime settings for your script in the configuration files. For details, see ["Configure Runtime Settings Manually" on page 305](#).

Use Bookmarks

When working in the Editor, VuGen lets you place bookmarks at various locations within your script. You can navigate between the bookmarks to analyze and debug your code. The following steps


describe how to work with bookmarks. Most of the bookmark functionality is available from VuGen's Bookmarks pane. To access the Bookmarks pane, click **View > Bookmarks**.

Create a Bookmark

In the Editor, place the cursor at the desired location and press Ctrl + F2. VuGen places a bookmark icon  in the left margin of the script.

Remove a Bookmark



To remove a bookmark, perform one of the following:

- In the Editor, click in the line that contains the bookmark and press Ctrl + F2.
- In the Bookmark pane, select the bookmark that you want to delete and click the **Delete Bookmark** button .

VuGen removes the bookmark icon from the left margin.

Navigate between Bookmarks

Click **View > Bookmarks** to display the Bookmarks pane.

- To move to the next bookmark, click the **Next Bookmark** button  or press **F2**.
- To return to the previous bookmark, click the **Previous Bookmark** button  or press **Shift + F2**.

You can navigate between bookmarks in the current action only. To navigate to a bookmark in another action, select that action in the left pane and then press F2.

Navigate to a Specific Bookmark in a Vuser Script

In the Bookmarks pane, double-click the specific bookmark to which you want to navigate. The cursor flashes in the Editor at the start of the line containing the bookmark.

Files Generated During Replay

This section describes what occurs when a Vuser script is replayed, and describes the files that are created.

1. The **options.txt** file is created. This file includes command line parameters to the preprocessor.

Example of options.txt file

```
-DCCI  
-D_IDA_XL
```

-DWINNT	
-Ic:\tmp\Vuser	(name and location of Vuser include files)
-IE:\LRUN45B2\include	(name and location of include files)
-ec:\tmp\Vuser\logfile.log	(name and location of output logfile)
c:\tmp\Vuser\VUSER.c	(name and location of file to be processed)

- The file **Vuser.c** is created. This file contains 'includes' to all the relevant .c and .h files.

Example of Vuser.c file

```
#include "E:\LRUN45B2\include\lrun.h"
#include "c:\tmp\web\init.c"
#include "c:\tmp\web\run.c"
#include "c:\tmp\web\end.c"
```

- The c preprocessor **cpp.exe** is invoked in order to 'fill in' any macro definitions, precompiler directives, and so on, from the development files.

The following command line is used:

```
cpp -foptions.txt
```

- The file **pre_cci.c** is created which is also a C file (**pre_cci.c** is defined in the **options.txt** file). The file **logfile.log** (also defined in **options.txt**) is created containing any output of this process. This file should be empty if there are no problems with the preprocessing stage. If the file is not empty then it is almost certain that the next stage of compilation will fail due to a fatal error.
- The **cci.exe** C compiler is now invoked to create a platform-dependent pseudo-binary file (.ci) to be used by the Vuser driver program that will interpret it at runtime. The cci takes the **pre_cci.c** file as input.
- The file **pre_cci.ci** is created as follows:

```
cci -errout c:\tmp\Vuser\logfile.log -c pre_cci.
```

- The file **logfile.log** is the log file containing output of the compilation.
- The file **pre_cci.ci** is now renamed to **Vuser.ci**.

Since the compilation can contain both warnings and errors, and since the driver does not know the results of this process, the driver first checks if there are entries in the **logfile.log** file. If there are, it then checks if the file **Vuser.ci** has been built. If the file size is not zero, it means that the cci has succeeded to compile - if not, then the compilation has failed and an error message will be given.

- The relevant driver is now run, taking both the **.usr** file and the **Vuser.ci** file as input. For example:

```
mdrv.exe -usr c:\tmp\Vuser\usr -out c:\tmp\Vuser -file
c:\tmp\Vuser\Vuser.ci
```

The **.usr** file is needed since it tells the driver program which database is being used. This determines which libraries need to be loaded for the run.

10. If there is an existing replay log file, **output.txt**, (see the following entry), the log file is copied to **output.bak**.
11. The **output.txt** file is created (in the path defined by the 'out' variable). This file contains the output messages that were generated during the script replay. These are the same messages that appear in the Replay view of VuGen's Output pane.

NV Insights Report

Note:

- The report is available for the following Vuser protocols only: Web HTTP/HTML, TruClient Web, Flex, SAP Web, and Siebel Web.

This section describes how to view an NV Insights Report in VuGen after running a Vuser script.

The NV Insights Report assists in pinpointing factors that negatively impact an application's performance across a network. The data in an NV Insights Report is derived from packet list data; the report displays the resulting data in an informative report that provides insight into an application's operation. The NV Insights Report consists of several sub-reports, each displaying different aspects of the network data captured during the replay of the Vuser script.

The report includes performance optimization recommendations and an HTTP analysis and resources breakdown, as well as load times, component download analysis, response time breakdown, and details of errors received.

Note: VuGen saves an HTML version of the NV Insights Report for the most recent replay in the following location: <script's folder>\result1\NVReport\EmbeddedReport.html

For details on the NV Insights Report, see the [Network Virtualization for LoadRunner & Performance Center Help](#).

Opening the NV Insights Report

1. Make sure the NV Insights Report is enabled.

Note: To enable/disable the NV Insights Report, in VuGen click **Tools > Options > Scripting > Replay**, and select/clear the **Display NV Insights Report** check box. This option is enabled by default when the NV for Load Generator and VuGen component is installed on the VuGen machine. For details, see "[Display NV Insights Report](#)" on page 99.

For details on how to install the required Network Virtualization component, see the [Network Virtualization for LoadRunner & Performance Center Help](#).

2. After a script has finished running, open the NV Insights Report by clicking the **Open NV Insights Report** link in VuGen's Replay Summary page (displayed after script replay).

Note: Generating the NV Insights Report may take time. Click **Cancel** (in VuGen's Replay Summary page) to abort report generation.

Limitations

- The NV Insights Report is not supported for WinINet replay mode (set for a Web Vuser in **Replay > Runtime Settings > Internet Protocol > Preferences > Advanced**). In some instances, it is possible to modify a script that requires WinINet to run, to enable the script to run without WinINet. Contact customer support for more information.
- The NV Insights Report cannot be created if several instances of VuGen are running at the same time.
- The NV Insights Report supports only IPv4 network traffic—not IPv6.
- The NV engine permits the following characters in transaction names: a-z|A-Z|0-9|.|#|_|-|'
Any other characters in transaction names are replaced by the character <-> in the NV Insights Report. (The original transaction name will still appear in the LoadRunner Analysis reports.)
- In order to distinguish transactions with the same name, during the merge process the NV engine drops all characters in the transaction name that appear after three consecutive underscores ("___"). Do not use three consecutive underscores in transaction names.

Debugging

The **Debugging** section describes the various methods that are available to debug Vuser scripts.

Debugging Overview

Developing a Vuser script includes the steps shown below. This topic provides an overview of the fifth step, debugging a Vuser script.



After creating a Vuser script, replay the script to verify that the script runs without errors. Using VuGen's debug features, you can identify and resolve errors in your scripts. You can access most of these script debugging features from the VuGen toolbar.




Running a Vuser script

To run a Vuser script until the end of the script or until the next breakpoint, perform one of the following:

- Select **Replay > Run**.

- Click the **Run** button  on the Vuser toolbar.
- Press **F5**.

Note: The status of the Vuser script execution appears in the lower left corner of VuGen. The script execution status may be **Ready**, **Running**, or **Paused**.

- To stop a script that is running, click the **Stop Replay**  button on the VuGen toolbar.
- To pause a script that is running, click the **Pause**  button on the VuGen toolbar.
- To continue running a script that is paused, click the **Continue**  button on the VuGen toolbar.

The Run Step by Step Command

The **Run Step by Step** command runs the script one line at a time. This enables you to follow the script execution. The **Run Step by Step** command starts the script replay, and then and pauses it on the first line of the script, usually in the `vuser_init()` action.

To run the script step by step, perform one of the following:

- Select **Replay > Run Step by Step**.
- Click the **Run Step by Step** button  on the VuGen toolbar.
- Press **F10**.

Note: The **Run Step by Step** button is available only while a script is being replayed.

Breakpoints

Breakpoints pause script execution at specified points in the script. This enables you to examine the effects of the script on your application at pre-determined points during script execution.

- For concept details on breakpoints, see ["Working with Breakpoints" on page 317](#).
- For task details, see ["Debug Scripts with Breakpoints" on page 320](#).

Bookmarks

When working in Script view, VuGen lets you place bookmarks at various locations within your script. You can navigate between the bookmarks to help analyze and debug your code.

- For task details, see ["Use Bookmarks" on page 309](#).

Watching Variables

The Watch pane enables you to monitor variables and expressions while a script runs. You can monitor variables and expressions only when execution of a Vuser script is in the Paused state. To display the Watch pane, click **View > Debug > Watch**. For details on using the Watch pane, see ["Watching Expressions and Variables" on page 318](#).

Go To Commands

- To navigate around a script using breakpoints, you can use the **Go To Source** command. For details, see ["Debug Scripts with Breakpoints" on page 320](#).
- To navigate around a script using bookmarks, you can use the **Next Bookmark** and **Previous Bookmark** commands. For details, see ["Use Bookmarks" on page 309](#).

If you want to examine the Replay log messages for a specific step or function, right-click the step in the Editor and select **Go To Step in Replay Log**. VuGen places the cursor at the start of the corresponding step in the Output pane's Replay log.

Output Pane

The Output pane displays messages that were generated during the replay of your script. For details, see ["Output Pane" on page 77](#).

Correlation

To enable some recorded Vuser scripts to replay correctly, it may be necessary to implement correlation. Correlation is used when a recorded script includes a dynamic value (such as a session ID) and therefore cannot be successfully replayed. To resolve this, you convert the dynamic value into a variable—thereby enabling your script to replay successfully. For details, see ["Correlation Overview" on page 232](#).

Error Handling

You can specify how a Vuser handles errors during script execution. By default, when a Vuser detects an error, the Vuser stops executing the script. You can instruct a Vuser to continue with the next iteration when an error occurs using one of the following methods:

- Using runtime settings. You can specify the **Continue on Error** runtime setting. The **Continue on Error** runtime setting applies to the entire Vuser script. You can use the `lr_continue_on_error` function to override the **Continue on Error** runtime setting for a portion of a script.
- Using the `lr_continue_on_error` function. The `lr_continue_on_error` function enables you to control error handling for a specific segment of a Vuser script. To mark the segment, enclose it with `lr_continue_on_error(1);` and `lr_continue_on_error(0);` statements. The new error settings apply to the enclosed Vuser script segment. See the paragraphs below for details.

For example, if you enable the Continue on Error runtime setting and a Vuser encounters an error during replay of the following script segment, the Vuser continues executing the script:

```
web_link("EBOOKS",
```

```
    "Text=EBOOKS",  
    "Snapshot=t2.inf",  
    LAST);  
web_link("Find Rocket eBooks",  
    "Text=Find Rocket eBooks",  
    "Snapshot=t3.inf",  
    LAST);
```

To instruct the Vuser to continue on error for a specific segment of the script, enclose the segment with the appropriate `lr_continue_on_error` statements:

```
lr_continue_on_error(1);  
    web_link("EBOOKS",  
        "Text=EBOOKS",  
        "Snapshot=t2.inf",  
        LAST);  
    web_link("Find Rocket eBooks",  
        "Text=Find Rocket eBooks",  
        "Snapshot=t3.inf",  
        LAST);  
lr_continue_on_error(0);
```

Additional Debugging Information

General Debugging Tip

VuGen can be used as a regular text editor. You can open any text file in it and edit it. When an error message is displayed during replay in the output window below, you can double click on it and VuGen jumps the cursor to the line of the test that caused the problem. You can also place the cursor on the error code and press F1 to view the online help explanation for the error code.

Using C Functions for Tracing

You can use the C interpreter trace option (in version 230 or higher) to debug your Vuser scripts. The `ci_set_debug` statement allows trace and debug to be turned on and off at specific points in the script.

```
ci_set_debug(ci_this_context, int debug, int trace);
```

For example, you could add the following statements to your script:

```
ci_set_debug(ci_this_context, 1, 1) /* turn ON trace =; debug */  
ci_set_debug(ci_this_context, 0, 0) /* turn OFF trace =; debug */
```

Additional C Language Keywords

When you run a C script in VuGen, its parser uses the built-in C interpreter to parse the functions in the script. You can add keywords that are not part of the standard parser's library. By default, several

common C++ keywords are added during installation, such as *size_t* and *DWORD*. You can edit the list and add additional keywords for your environment.

Add Additional Keywords

1. Open the `vugen_extra_keywords.ini` file, located in your machine's <Windows> or <Windows>/System directory.
2. In the `EXTRA_KEYWORDS_C` section, add the desired keywords for the C interpreter.

The file has the following format:

```
[EXTRA_KEYWORDS_C]
FILE=
size_t=
WORD=
DWORD=
LPCSTR=
```

Examining Replay Output

Look at the replay output (either from within VuGen, or the file `output.txt` representing the output of the VuGen driver). You may also change the runtime settings options in VuGen to select more extensive logging in order to obtain a more detailed log output of the replayed test.

Working with Breakpoints

VuGen lets you include breakpoints in your Vuser scripts to help you to debug the scripts. Breakpoints pause script execution at specified points in the script. This enables you to analyze the effects of the script on your application at pre-determined points during script execution. For task details, see ["Debug Scripts with Breakpoints" on page 320](#). A breakpoint symbol (●) in the left margin of the script

indicates the presence of a breakpoint. In addition, VuGen highlights the line in the script.

You can disable a breakpoint if the breakpoint is temporarily not required. A white dot inside the Breakpoint symbol indicates a disabled breakpoint (◐). When a breakpoint is disabled, script execution

continues at the disabled breakpoint and is paused at the following enabled breakpoint. You use the Breakpoints pane to enable and disable breakpoints. In addition, the breakpoints pane enables you to delete an existing breakpoint or delete all existing breakpoints. To display the Breakpoints pane, click **View > Debug > Breakpoints**.

To run a script with breakpoints, begin running the script as usual. VuGen pauses script execution when it reaches a breakpoint. You can examine the effects of the script run up to the breakpoint, make any necessary changes, and then restart the script from the breakpoint.





To resume execution, select **Replay > Run**. Once restarted, the script continues until it encounters another breakpoint or the end of the script.

Breakpoints Pane

This VuGen pane enables you to set and manage breakpoints to help analyze the effects of the script on your application at pre-determined points during script execution.

To access	View > Debug > Breakpoints
Important information	<ul style="list-style-type: none"> This pane is relevant only when a run session is paused. You can move this pane to different areas of the Main User Interface. For details, see "How to Modify the VuGen Layout" on page 40.
See also	<ul style="list-style-type: none"> "VuGen User Interface" on page 37 "Debugging" on page 313

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Enables you to delete the selected breakpoint.
	Enables you to enable or disable the selected breakpoint.
	Enables you to navigate to a specific breakpoint in a Vuser script.
	Deletes all breakpoints in the Vuser script.
Enabled	A check box that specifies whether the breakpoint is enabled or disabled, and enables you to enable or disable the adjacent breakpoint.
Name	The name of the file that contains the breakpoint, and the line number within the file that contains the breakpoint.
Script	The name of the Vuser script that contains the breakpoint.
Function Name	The name of the function within the Vuser script that contains the breakpoint.

Watching Expressions and Variables

VuGen's Watch pane enables you to monitor variables while a script runs. The list of variables that you want to watch is known as the watch list, and is displayed in the watch pane. To display the Watch pane, click **View > Debug > Watch**. You can add only variables to the watch list - you cannot add expressions to the watch list. You can add, edit, or remove variables within the watch list by using the Watch pane's


toolbar buttons. You can sort the columns in the watch pane by expression, value, or type name by clicking the column headers. For details on other debugging features in VuGen, see ["Debugging Overview" on page 313](#).

Note: You can monitor variables only when execution of a Vuser script is in the **Paused** state.

Adding a New Watch to the Watch List

You can add a new watch expression only when execution of a Vuser script is in the **Paused** state.

To add a new watch:


1. Click **View > Debug > Watch** to open the Watch pane.
2. Click the **Add Watch** button . The Add New Watch dialog box opens.
3. In the **Expression** field, enter the variable that you want to watch, and then click **OK**. VuGen adds the variable to the list of expressions in the watch list.

Note: You can add only variables to the watch list - you cannot add expressions to the watch list.

Editing a Watch Expression

Note: You can edit a watch expression only when execution of a Vuser script is in the **Paused** state.


To edit a watch expression:

1. Click **View > Debug > Watch** to open the Watch pane.
2. In the watch list, select the expression that you want to edit, and then click the **Edit Watch Expression** button . The Edit Watch dialog box opens.
3. In the **Expression** field, modify the existing variable as required, and then click **OK**. VuGen displays the modified variable in the list of variables in the watch list.

Deleting a Watch Expression

Note: You can delete a watch expression only when execution of a Vuser script is in the **Paused** state.


To delete a watch expression:

1. Click **View > Debug > Watch** to open the Watch pane.
2. In the Watch pane, select the expression that you want to delete, and then click the **Delete Watch** button . VuGen deletes the selected expression from the list of expressions in the watch list.

Deleting All Watch Expressions From the Watch List

Note: You can delete watch expressions only when execution of a Vuser script is in the **Paused** state.

To delete all watch expressions from the watch list:

1. Click **View > Debug > Watch** to open the Watch pane.
2. Click the **Delete All Watches** button . VuGen deletes all the expressions from the watch list.

Debugging Web Vuser Scripts

VuGen provides an additional tool to help you debug Web Vuser scripts—a runtime viewer. You can instruct VuGen to display a runtime viewer when you run a Web Vuser script. The runtime viewer was developed specifically for use with VuGen—it is unrelated to the browser that you use to record your Vuser scripts.


The runtime viewer shows each Web page as it is accessed by the Vuser. This is useful when you debug Web Vuser scripts because it allows you to check that the Vuser accesses the correct Web pages. For information on how to enable the viewer, see ["Scripting Options Tab" on page 96](#).

Note: The runtime viewer, in order to conserve resources, may display part of the page's HTML as text.

Debug Scripts with Breakpoints

The following describes how to work with breakpoints.

To	Do this
Add/Remove a breakpoint	<p>Locate the cursor in the script where you want to insert the breakpoint and then do one of the following:</p> <ul style="list-style-type: none"> • Select Replay > Toggle Breakpoint. • Press F9. • Click in the left margin of the script, adjacent to where you want to insert the breakpoint. <p>Add: The Breakpoint symbol (●) appears in the left margin of the script, and VuGen highlights the line in the script.</p> <p>Remove: The Breakpoint symbol (●) is removed from the left margin of the script.</p>
Enable/Disable a breakpoint	<ol style="list-style-type: none"> 1. Click View > Debug > Breakpoints to display the Breakpoints pane. 2. Select the appropriate Enable check box to enable a breakpoint. The Breakpoint symbol (●) appears in the left margin of the script. 3. Clear the appropriate Enable check box to disable a breakpoint. The Disabled Breakpoint symbol (○) appears in the left margin of the script. <p>When a breakpoint is disabled, script execution continues at the disabled breakpoint and is paused at the following enabled breakpoint.</p>

To	Do this
Manage breakpoints	The Breakpoints pane allows you to remove, enable, and disable breakpoints in a Vuser script. For user interface details, see "Breakpoints Pane" on page 78 .
Navigate to a specific breakpoint in a Vuser script	<ul style="list-style-type: none"> In the Breakpoints pane, select the specific breakpoint to which you want to navigate, and then click the Go to source button . In the Breakpoints pane, double-click the breakpoint to which you want to navigate. <p>The cursor flashes in the Editor at the start of the line containing the breakpoint.</p>
Run a script with breakpoints	<p>Begin running the script as usual. VuGen pauses script execution when it reaches a breakpoint. You can examine the effects of the script run up to the breakpoint, make any necessary changes, and then restart the script from the breakpoint.</p> <p>To resume execution, select Replay > Run. Once restarted, the script continues until it encounters another breakpoint or the end of the script.</p>

See also:

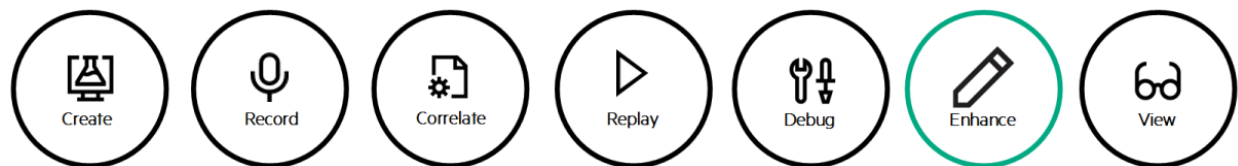
- Overview: ["Working with Breakpoints" on page 317](#).

Enhancing

The **Enhancing** section explains the features that VuGen provides to enable you to enhance Vuser scripts that will be able to accurately generate load. This includes features such as transactions and rendezvous points.

Enhancing a Script for Load Testing - Overview

Developing a Vuser script includes the steps shown below.



This section describes optional tasks for preparing a Vuser script for load testing.

Add Parameterization

When you record a business process, VuGen generates a script that contains the actual values used during recording. Suppose you want to perform the script's actions (query, submit, and so forth) using different values from those recorded. To do this, you replace the recorded values with parameters. This is known as *parameterizing* the script. For details, see ["Parameterization Overview" on page 342](#).

Insert Transactions

You can insert transactions into your Vuser script either while recording the script or after recording the script.


For inserting transactions during recording, use the buttons on the floating toolbar, or click **Ctrl + T**. For inserting transactions into your script after recording, use the **Design > Insert in Script** menu items.

For task details, see ["Insert Transactions" on page 324](#).

Insert Rendezvous Points

You can instruct multiple Vusers to perform a task at exactly the same moment using a rendezvous point. When a Vuser arrives at the rendezvous point, it waits until all Vusers participating in the rendezvous arrive. When the designated number of Vusers arrive, the Vusers are released.

You can insert rendezvous points in one of the following ways:

- To insert a rendezvous point while recording, click the **Rendezvous** button  on the Recording toolbar and enter a name in the dialog box (not case sensitive).
- To insert a rendezvous point after recording, select **Design > Insert in Script > Rendezvous** and enter a name for the rendezvous point (not case sensitive).

When a rendezvous point is inserted, VuGen inserts a **lr_rendezvous** function into the Vuser script. For example, the following function defines a rendezvous point named rendezvous1:

```
lr_rendezvous("rendezvous1");
```

For concept details, see ["Rendezvous Points" on page 328](#).

Insert VuGen Functions

You can insert VuGen functions at this point. For a list of some useful functions see ["Adding VuGen Functions Overview" on page 328](#).

Insert Steps

You can insert a variety of steps into your script such as think time steps, debug messages, and output messages. For task details, see ["Insert Steps into a Script" on page 340](#).

Insert Comments

VuGen allows you to insert comments between Vuser activities. You can insert a comment to describe an activity or to provide information about a specific operation. For example, if you are recording database actions, you could insert a comment to mark the first query, such as "This is the first query."

You can insert a comment in one of the following ways:

- To insert a comment while recording, click the **Insert Comment** button  on the Recording toolbar and enter the desired comment in the Insert Comment dialog box.

- To insert a comment after recording, select **Design > Insert in Script > Comment** and enter the comment.

The following script segment shows how a comment appears in a Vuser script:

```
/* <comments> */
```

Insert Log Messages

You can use VuGen to generate and insert **lr_log_message** functions into a Vuser script. For example, if you are recording database actions, you could insert a message to indicate the first query, "This is the first query."

To insert a log message, select **Design > Insert in Script > Log Message** and enter the message.

Insert Synchronization Points (RTE Vusers only)

You can add synchronization functions to synchronize the execution of the Vuser script with the output from your application. Synchronization applies to RTE Vuser scripts only.

The following is a list of the available synchronization functions:

Function	Description
TE_wait_cursor	Waits for the cursor to appear at a specified location in the terminal window.
TE_wait_silent	Waits for the client application to be silent for a specified number of seconds.
TE_wait_sync	Waits for the system to return from X-SYSTEM or Input Inhibited mode.
TE_wait_text	Waits for a string to appear in a designated location.
TE_wait_sync_transaction	Records the time that the system remained in the most recent X-SYSTEM mode.

For details about synchronization in RTE Vuser scripts, see ["RTE Synchronization Overview" on page 625](#).



See also:

- ["Replaying" on page 277](#)

Transaction Overview

You define *transactions* to measure the performance of the server. Each transaction measures the time it takes for the server to respond to specified Vuser requests. These requests can be simple tasks such

as waiting for a response for a single query, or complex tasks, such as submitting several queries and generating a report.

To measure a transaction, you insert Vuser functions to mark the beginning and the end of a task. Within a script, you can mark an unlimited number of transactions, each transaction with a different name.

For LoadRunner, the Controller measures the time that it takes to perform each transaction. After the test run, you analyze the server's performance per transaction using the Analysis' graphs and reports.



Before creating a script, you should determine which business processes you want to measure. You then mark each business process or sub-process as a transaction.

Avoid using a "," or "@" symbol in a transaction name. These characters may cause errors to occur when attempting to open the Analysis Cross Results graphs.

You can create transactions either during or after recording. For task details, see ["Insert Transactions" below](#).

Insert Transactions

Insert a transaction while recording

To	Do this
Mark the start of a transaction	<p>On the Recording toolbar, click the Start Transaction button  , enter a transaction name, and click OK.</p> <p>When the script is generated, VuGen inserts an lr_start_transaction statement into the Vuser script.</p>
Mark the end of a transaction	<p>On the Recording toolbar, click the End Transaction button  and select the transaction to close.</p> <p>When the script is generated, VuGen inserts an lr_end_transaction statement into the Vuser script.</p>

Insert a transaction after recording

You use VuGen's Editor to insert a transaction after recording a Vuser script.

To	Do this
Mark the start of a transaction	<p>Locate the cursor in the script where you want to start the transaction and do one of the following:</p> <ul style="list-style-type: none"> • Select Design > Insert in Script > Start Transaction. • Press Ctrl+T. • Right-click in the script where you want to start the transaction and select Insert > Start Transaction. <p>VuGen inserts an lr_start_transaction statement into the Vuser script. Enter a transaction name into the new step.</p>
Mark the end of a transaction	<p>Locate the cursor in the script where you want to end the transaction and do one of the following:</p> <ul style="list-style-type: none"> • Select Design > Insert in Script > End Transaction. • Press Ctrl+Shift+T. • Right-click in the script where you want to end the transaction and select Insert > End Transaction. <p>VuGen inserts an lr_end_transaction statement into the Vuser script. Enter a transaction name into the new step.</p>
Simultaneously mark the start and end of a transaction	<ol style="list-style-type: none"> 1. Select the steps that you want to include in the transaction. 2. Do one of the following: <ul style="list-style-type: none"> • Select Design > Insert in Script > Surround with Transaction. • Press Shift+Ctrl+I. • Right-click inside the selection and select Surround with Transaction. <p>The Surround with Transaction dialog box opens.</p> 3. Enter a name for the transaction and click OK. <p>VuGen inserts an lr_start_transaction statement before the first selected step, and an lr_end_transaction statement after the last selected step.</p>

Transaction guidelines

- You can create **nested** transactions—transactions within transactions. If you nest transactions, close the inner transactions before closing the outer ones—otherwise the transactions cannot be analyzed properly. Nested transactions must be contained within a single **action** section.
- Transaction names must be unique and may contain letters or numbers. Do not use the following characters: ., : # / \ " & ' (single quote)
- A failed transaction does not automatically set the script's Replay status to Failed.

See also:

- ["Transaction Overview" on page 323](#)

Display Transactions

The following steps describe how to display different types of transactions when viewing them in the task pane. For background information, see ["Transaction Overview" on page 323](#).

Display Hidden Transactions

To display the hidden transactions—the non-primary and client side transactions—click the button adjacent to **Show hidden transactions** at the bottom of the transaction list. VuGen lists the hidden transactions in gray. To hide them, click the button again.

Display Transactions With Errors

Transactions with errors are those that do not measure any server steps, or those with illegal names. To show the transactions with errors, click the **Show transactions with errors** button. VuGen lists the transactions with errors in red. To hide them, click the button again.

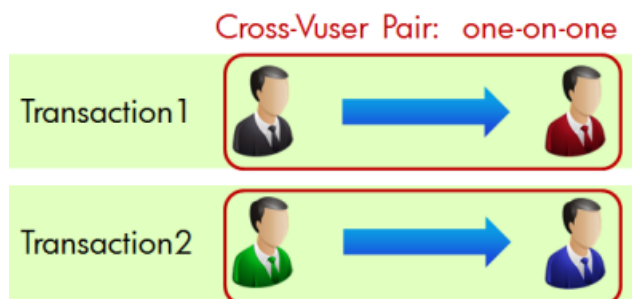
Display Transactions for Non-primary Steps

To show the transactions for non-primary steps, you need to display all of the thumbnails. Select **View > Show All Thumbnails**.

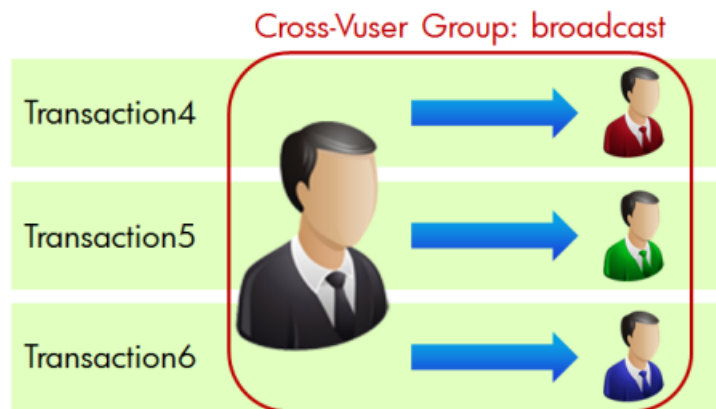
Cross-Vuser Transaction Overview

Cross-Vuser transactions are transactions that allow you to measure the duration of a process that involves multiple Vusers. For example, you can create a *cross-Vuser transaction* to determine how long it took the receiving party to get the information that was sent. This transaction type originates from one Vuser and ends at another.

The Vusers who start and end the transaction, form a *cross-Vuser pair*.



A cross-Vuser transaction is not limited to two Vusers. It also includes broadcasting, in which one Vuser sends a message to many Vusers. In this case, the broadcaster, who begins the transaction, and the receivers, who end the transaction, together form a *cross-Vuser group*.



For both a cross-Vuser pair and a cross-Vuser group, the transaction is initiated by a single Vuser.

You must define a transaction ID to serve as an identifier for each cross-Vuser pair or group. The ID must be a string that uniquely identifies the pair or group. All Vusers in a pair or group share the same identifier.

It is recommended that you create a standard for the transaction ID. The ID string should indicate whether the cross-Vuser transaction is related to a pair or group.

In the above example, in Transaction1, if the sender is "black", the receiver is "red", and the message is "message1", then a logical ID string could be *black_red_message1*. In Transaction4, if the sender is "black", then a logical ID string could be *black_broadcast*.

The following examples illustrate typical scenarios for cross-Vuser transactions:

- The time it took from when one user sent an email to when another user received it.
- The time it took for users to see a message posted on a social network.

Use the following guidelines when creating a script with cross-Vuser transactions:

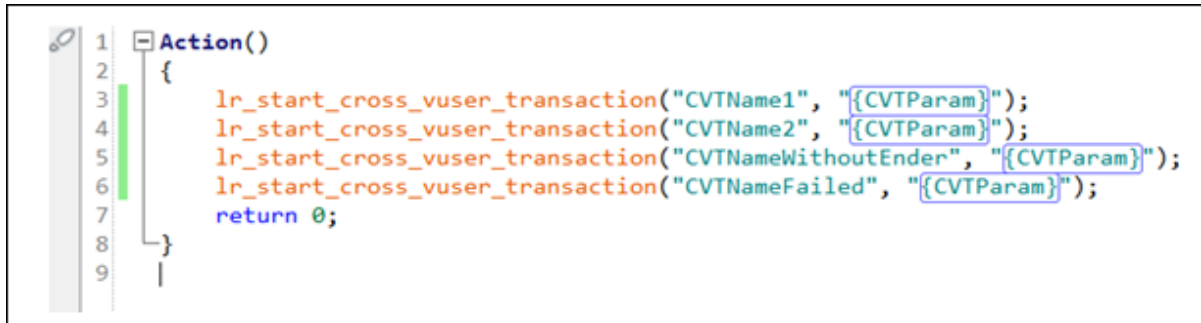
- Cross-Vuser transactions do not calculate think time, waste time, and so forth—only duration time is recorded.
- Cross-Vuser transaction data is not used by the Controller's transaction monitors.
- LoadRunner cannot detect the status of cross-Vuser transactions.

For details on how to create cross-Vuser transactions, see ["Create a Cross-Vuser Transaction" below](#).

Create a Cross-Vuser Transaction

Cross-Vuser transactions are transactions that allow you to measure the duration of a process that involves multiple Vusers.

1. Open the Steps Toolbox (from VuGen's **View** menu) and manually add **lr_start_cross_vuser_transaction** functions at the beginning of the transactions.



```
1 Action()  
2 {  
3     lr_start_cross_vuser_transaction("CVTName1", "{CVTParam}");  
4     lr_start_cross_vuser_transaction("CVTName2", "{CVTParam}");  
5     lr_start_cross_vuser_transaction("CVTNameWithoutEnder", "{CVTParam}");  
6     lr_start_cross_vuser_transaction("CVTNameFailed", "{CVTParam}");  
7     return 0;  
8 }  
9
```

2. Drag in **lr_end_cross_vuser_transaction** functions to your script to mark the end of the transactions.
3. Fill in the **Transaction name** and **Transaction ID** fields. Make sure you create a unique ID for your transaction. For guidelines, see the ["Cross-Vuser Transaction Overview" on page 326](#). Note that you can parameterize the transaction ID as any other standard parameter. For details, see ["Create Parameters" on page 347](#).
4. Replay the script in VuGen to check its functionality. Check the Output log for any error messages.

See also:

- ["Cross-Vuser Transaction Overview" on page 326](#)

Rendezvous Points

When performing load testing, you need to emulate heavy user load on your system. To accomplish this, you instruct Vusers to perform a task at exactly the same moment using a rendezvous point. When a Vuser arrives at the rendezvous point, it waits until all Vusers participating in the rendezvous arrive. When the designated number of Vusers arrive, they are released.

You designate the meeting place by inserting a rendezvous point into your Vuser script. When a Vuser executes a script and encounters the rendezvous point, script execution is paused and the Vuser waits for permission from the Controller to continue. After the Vuser is released from the rendezvous, it performs the next task in the script.

For task details, see ["Enhancing a Script for Load Testing - Overview" on page 321](#).

Note: Rendezvous points are effective only in Action sections—not init or end sections.

Adding VuGen Functions Overview

This section contains useful VuGen functions that you may want to add to your script while debugging or preparing your script for load testing.

Obtain Vuser Information

You can add the following functions to your Vuser scripts to retrieve Vuser information:

Function	Description
lr_get_attrib_string	Returns a command line parameter string.
lr_get_host_name	Returns the name of the machine running the Vuser script.
lr_get_master_host_name	Returns the name of the machine running the Controller. Not applicable when working with HPE Application Performance Management.
lr_whoami	Returns the name of a Vuser executing the script. Not applicable when working with HPE Application Performance Management.

In the following example, the **lr_get_host_name** function retrieves the name of the computer on which the Vuser is running.

```
my_host = lr_get_host_name( );
```

For more information about the above functions, see the [Function Reference \(Help > Function Reference\)](#).

Send Messages to Output

Using the Message type functions in your Vuser script, you can send customized error and notification messages to the output and log files, and to the Test Report summary. For example, you could insert a message that displays the current state of the client application. You can also save these messages to a file.

When working with HPE Application Performance Management, you can use Message type functions to send error and notification messages to the Web site or Business Process Monitor log files. For example, you could insert a message that displays the current state of the Web-based application.

Note: Do not send messages from within a transaction as this may lengthen the transaction execution time and skew the transaction results.

You can use the following message functions in your Vuser scripts:

Function	Description
lr_debug_message	Sends a debug message to the Output window or the Business Process Monitor log file.
lr_error_message	Sends an error message to the Output window or the Business Process Monitor log files.

lr_get_debug_message	Retrieves the current message class.
lr_log_message	Sends an output message directly to the log file, <i>output.txt</i> , located in the Vuser script folder. This function is useful in preventing output messages from interfering with TCP/IP traffic.
lr_output_message	Sends a message to the Output window or the Business Process Monitor log files.
lr_set_debug_message	Sets a message class for output messages.
lr_vuser_status_message	Sends a message to the Vuser status area in the Controller. Not applicable when working with HPE Application Performance Management.
lr_message	Sends a message to the Vuser log and Output window or the Business Process Monitor log files.

The behavior of the **lr_message**, **lr_output_message**, and **lr_log_message** functions are not affected by the script's debugging level in the Log runtime settings—they will always send messages.

General Vuser Functions

The general Vuser functions are also called LR functions because each LR function has an **lr** prefix. The LR functions can be used in any type of Vuser script. The LR functions enable you to:

- Get runtime information about a Vuser, its Vuser Group, and its host.
- Add transactions and synchronization points to a Vuser script. For example, the **lr_start_transaction** (**lr.start_transaction** in Java) function marks the beginning of a transaction, and the **lr_end_transaction** (**lr.end_transaction** in Java) function marks the end of a transaction. See ["Enhancing a Script for Load Testing - Overview" on page 321](#) for more information.
- Send messages to the output, indicating an error or a warning.



See also:

- Function Reference (**Help > Function Reference**)

Protocol-Specific Vuser Functions

In addition to the general Vuser functions, VuGen also generates and inserts protocol-specific functions into the Vuser script while you record.

The protocol-specific functions are particular to the type of Vuser that you are recording. For example, VuGen inserts LRS functions into a Windows Sockets script.

By default, VuGen's automatic script generator creates Vuser scripts in C for most protocols, and in Java for Java type protocols. You can instruct VuGen to generate code in Visual Basic or Javascript. For more information, see ["General > Script Recording Options" on page 169](#).

All standard conventions apply to the scripts, including control flow and syntax. You can add comments and conditional statements to the script just as you do in other programming languages.

The following segment from a Web Vuser script shows several functions that VuGen recorded and generated in a script:

```
#include "as_web.h"
Action1()
{
    web_add_cookie("nav=140; DOMAIN=dogbert");
    web_url("dogbert",
        "URL=http://dogbert/",
        "RecContentType=text/html",
        LAST);
    web_image("Library",
        "Alt=Library",
        LAST);
    web_link("1 Book Search:",
        "Text=1 Book Search:",
        LAST);
    lr_start_transaction("Purchase_Order");
    ...
}
```

See also:

- C functions: Function Reference (**Help > Function Reference**)

Note: The C Interpreter used for running Vuser scripts written in C only supports the ANSI C language. It does not support the Microsoft extensions to ANSI C.

- Java: ["Java Vuser Protocol" on page 536](#)

Encoding Passwords and Text

You can mask and unmask text in your script to hide or show your passwords and other confidential text strings. For example, you may want to hide personal user information, such as social security or ID numbers.

To enable the script to use a masked string:

Call the **lr_unmask** function, for example:

```
lr_start_transaction(lr_unmask("3c29f4486a595750"));
```

VuGen uses the original string value while displaying an encoded (masked) string in the script.

To mask a string:

1. Select the text you want to mask.
2. Right-click and select **Mask string** from the context menu.
VuGen masks the visible string and adds an **lr_mask** function call.

To unmask a string:

1. Select the string you want to unmask.
2. Right-click and select **Restore masked string (string)** from the context menu.

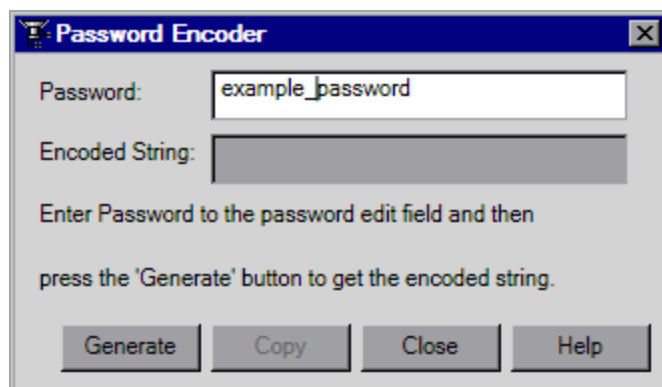
See also:

- For more information on the **lr_unmask** function, see the [Function Reference \(Help > Function Reference\)](#).

Password Encoder

This tool enables you to generate masked passwords that you can use as arguments in your script or as parameter values.

For example, your Web site may include a form in which the user must supply a password. You may want to test how your site responds to different passwords, but you also want to protect the integrity of the passwords by hiding the plain-text password values.



To access and use

1. On the LoadRunner machine, select **Start > All Programs > HPE Software > HPE LoadRunner > Tools > Password Encoder**.
2. Enter the password in the **Password** box.
3. Click **Generate**. The Password Encoder masks the password and displays it in the **Encoded String** box.
4. Click **Copy** and paste the encoded value in the script.

User interface elements are described below:

UI Element	Description
Copy	Copy the results from the encoded string field to paste them to the Data table containing your list of parameters.
Encoded String	The encoded results are displayed here.
Generate	Click this to generate the encoded password.
Password	Enter the password you want to encode here.

See also:

- ["Encoding Passwords and Text" on page 331](#)

Masking Text

You can mask and unmask text in your script to hide or show your passwords and other confidential text strings.

To enable the script to use a masked string:

Call the **lr_unmask** function, for example:

```
lr_start_transaction(lr_unmask("3c29f4486a595750"));
```

VuGen uses the original string value while displaying an encoded (masked) string in the script.

To mask a string:

1. Select the text you want to mask.
2. Right-click and select **Mask string** from the context menu.
VuGen masks the visible string and adds an **lr_mask** function call.

To unmask a string:

1. Select the string you want to unmask.
2. Right-click and select **Restore masked string (string)** from the context menu.

See also:

- For more information on the **lr_unmask** function, see the [Function Reference \(Help > Function Reference\)](#).

Database Integration Overview

When testing your application or Web service, it is vital that you use data that is accurate and up to date. If you use a snapshot of data from a past date, it may no longer be valid or relevant.

The database integration allows you to access values in a database during your test, ensuring that the data is up to date. You can also check your returned values against those in the database.

The following is a list of the database functions, available from the **Database** category in the Steps Toolbox:

lr_db_connect	Connects to a database.
lr_db_disconnect	Disconnects from a database.
lr_db_executeSQLStatement	Submits an SQL statement to a database.
lr_db_dataset_action	Performs an action on a dataset.
lr_db_getValue	Retrieves a value from a dataset.
lr_db_dataset_action	Validates database contents by setting checkpoints.

The database integration functions are useful in the following scenarios:

- ["Connecting to a Database" below](#)
- ["Using Data Retrieved from SQL Queries" on the next page](#)
- ["Validating Database Values" on page 337](#)
- ["Checking Returned Values Through a Database" on page 338](#)
- ["Performing Actions on Datasets" on page 339](#)

For more information, see the [Function Reference \(Help > Function Reference\)](#).

Connecting to a Database

To connect to a database, you add a connection step, **lr_db_connect**, to your script through the Steps Toolbox. A built-in **Connection String Generator** guides you in creating a connection string specific to your database and credentials. You can also test your connection before inserting the step.

When running your script with iterations, virtual users only repeat the **Action** section of the script. If you include the database connection step in the **Action** section, the test will repeat it for each iteration. Virtual Users only repeat the **Action** section of the script, but not the **vuser_init** or **vuser_end** sections. Therefore, we recommend that you place the database connection step in the **vuser_init** section, and the disconnect step, **lr_db_disconnect** in the **vuser_end** section.

In cases where you only need to do one query and scroll through the data, you should also place the query statements in the **vuser_init** section.

For additional tips for working with Web services, see ["Send Messages over JMS" on page 764](#).

Using Data Retrieved from SQL Queries

A normal use of database steps is fetching data from the database and using it at a later point in the script. Since the script retrieves the data during each test run, the data is up to date and relevant.

The following example illustrates a typical flow for a Web Service protocol script. A similar sequence can also be applied to other protocols.

Step	API function
Connect to database	lr_db_connect
Execute an SQL query	lr_db_executeSQLStatement
Retrieve and save the data	lr_db_getvalue to <param_name>
Web Service call	web_service_call with {<param_name>}
Disconnect from database	lr_db_disconnect

You can iterate through the results in two ways:

- save them to a simple parameter during each iteration
- use VuGen built-in iterations to scroll through the data

For more information, see the [Function Reference \(Help > Function Reference\)](#).

In the following Web service example, the **vuser_init** section connects to the database and performs a database query.

```
vuser_init()
{
  lr_db_connect("StepName=myStep",
    "ConnectionString=Initial Catalog=MyDB;Data Source=mylab.net;user id =sa
;password = 12345;" ,
    "ConnectionName=MyConnection",
    "ConnectionType=SQL",
    LAST);
  lr_db_executeSQLStatement("StepName=MyStep",
    "ConnectionName=MyConnection",
    "SQLQuery=SELECT * FROM Addresses",
    "DatasetName=ds1",
    LAST);
  return 0;
}
```

At the end of your test, disconnect from the database in the **vuser_end** section.

```
vuser_end()
{
```

```

lr_db_connect("StepName=myStep",
    "ConnectionString=Initial Catalog=MyDB;Data Source=LAB1.devlab.net;user id
=sa ;password = soarnd1314;" ,
    "ConnectionName=MyConnection",
    "ConnectionType=SQL",
    LAST);
return 0;
}

```

In the Action section, you include the steps to repeat. Note the use of the **Row** argument. In the first call to the database, you specify the first row with **Row=next**. To retrieve another value in the same row, use **current**.

```

Action()
{
    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=Name",
        "Row=next",
        "OutParam=nameParam",
        LAST);
    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=city",
        "Row=current",
        "OutParam=cityParam",
        LAST);
    /* Use the values that you retrieved from the database in your Web Service call */
    web_service_call( "StepName=EchoAddr_101",
        "SOAPMethod=SanityService|SanityServiceSoap|EchoAddr",
        "ResponseParam=response",
        "Service=SanityService",
        "ExpectedResponse=SoapResult",
        "Snapshot=t1227168459.inf",
        BEGIN_ARGUMENTS,
        "xml:addr="
            "<addr>"
                "<name>{nameParam}</name>"
                "<street></street>"
                "<city>{cityParam}</city>"
                "<state></state>"
                "<zip></zip>"
            "</addr>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);
    return 0;
}

```

Validating Database Values

In this use case, a test executes an action that modifies a database. The goal of this use case is to validate that the resulting values in the database are correct.

The following table shows a typical flow for a Web Services of the script. You can use a similar validation for other protocols.

Step	API function
Connect to database	lr_db_connect (in vuser_init section)
Web Service call	web_service_call
Execute an SQL query	lr_db_executeSQLStatement
Retrieve and save the data	lr_db_getvalue to <param_name>
Check the data	lr_checkpoint
Disconnect from database	lr_db_disconnect (in vuser_end section)

For more information, see the [Function Reference \(Help > Function Reference\)](#).

The following example illustrates this process of checking the data:

```

Action()
{
/* A Web Service call that modifies a database on the back end. */
web_service_call( "StepName=addAddr_102",
    "SOAPMethod=Axis2AddrBookService|Axis2AddrBookPort|addAddr",
    "ResponseParam=response",
    "Service=Axis2AddrBookService",
    "ExpectedResponse=SoapResult",
    "Snapshot=t1227169681.inf",
    BEGIN_ARGUMENTS,
    "xml:arg0="
        "<arg0>"
            "<name>{Customers}</name>"
            "<city>{City}</city>"
        "</arg0>",
    END_ARGUMENTS,
    LAST);
/* Query the database by the customer name that was modified by the Web Service*/
lr_db_executeSQLStatement("StepName=MyStep",
    "ConnectionName=MyConnection",
    "SQLQuery=SELECT * FROM Addresses WHERE name = '{Customers}' ",
    "DatasetName=ds1",
    LAST);
/* Get the values retrieved by the database query. */

```

```

lr_db_getvalue("StepName=MyStep",
    "DatasetName=ds1",
    "Column=Name",
    "Row=current",
    "OutParam=CustomerName",
    LAST);
/* Compare the actual value with the expected value stored in the database. */
lr_checkpoint("StepName=validateCustomer",
    "ActualValue={Customers}",
    "ExpectedValue={CustomerName}",
    "Compare=Equals",
    "StopOnValidationError=false",
    LAST);
return 0;
}

```

Checking Returned Values Through a Database

In this scenario, a user executes an action which returns a response. The goal of this scenario is to validate the response against expected values.

The expected values are stored in a database. The script fetches the expected results from a database and then compares them with the actual response.

The following table shows a typical flow of a Web Service protocol script. You can employ a similar flow for other protocols.

Step	API function
Connect to database	lr_db_connect (in vuser_init section)
Web Service call	web_service_call with Result=<result_param>
Execute an SQL query	lr_db_executeSQLStatement
Retrieve the expected data	lr_db_getvalue to <param_name>
Validate the data	soa_xml_validate with an XPATH checkpoints.
Disconnect from database	lr_db_disconnect (in vuser_end section)

The following example illustrates a typical validation of data returned by a Web Service call. The validation step compares the actual expected results:

```

Action()
{
    web_service_call( "StepName=GetAddr_102",
        "SOAPMethod=AddrBook | AddrBookSoapPort | GetAddr",
        "ResponseParam=response",

```

```
"Service=AddrBook",
"ExpectedResponse=SoapResult",
"Snapshot=t1227172583.inf",
BEGIN_ARGUMENTS,
"Name=abcde",
END_ARGUMENTS,
BEGIN_RESULT,
END_RESULT,
LAST);
lr_db_executeSQLStatement("StepName=MyStep",
"ConnectionName=MyConnection",
"SQLQuery=SELECT * FROM Addresses WHERE name = 'abcde' ",
"DatasetName=ds1",
LAST);
lr_db_getvalue("StepName=MyStep",
"DatasetName=ds1",
"Column=Name",
"Row=current",
"OutParam=CustomerName",
LAST);
soa_xml_validate ("StepName=XmlValidation_1146894916",
"Snapshot=t623713af7a594db2b5fef43da68ad59d.inf",
"XML={GetAddrAllArgsParam}",
"StopOnValidationError=0",
BEGIN_CHECKPOINTS,
CHECKPOINT,"XPATH=/*[local-name(.)='GetAddr'] [1]/*[local-name(.)
='Result' ] [1]/*[local-name(.)='name' ] [1]", "Value_Equals={CustomerName}",
END_CHECKPOINTS,
LAST);
return 0;
}
```

For more information, see the [Function Reference \(Help > Function Reference\)](#).

Performing Actions on Datasets

VuGen lets you perform actions on datasets returned by SQL queries.

The **lr_db_dataset_action** function performs the following actions on datasets:

- **Reset.** Set the cursor to the first record of the dataset.
- **Remove.** Releases the memory allocated for the dataset.
- **Print.** Prints the contents of the entire dataset to the Replay Log and other test report summaries.

Note: When you retrieve binary data through **lr_db_getvalue**, you cannot print its contents using the **Print** action.

For information about the syntax and usage of this function, see the [Function Reference \(Help > Function Reference\)](#).

Create a Controller Scenario from VuGen

Note: The following section only applies to LoadRunner. For information on integrating scripts into Business Process profiles, see the HPE Application Performance Management documentation.

In addition to creating scenarios from the LoadRunner Controller, you can also create a basic scenario from within VuGen. This is useful after you have created and tested your script and want to include it in a scenario.

To create this type of scenario, select **Tools > Create Controller Scenario** and complete the dialog box. For user interface details, see ["Create Controller Scenario Dialog Box" on the next page](#).

For more information, see the HPE Controller User Guide.

Insert Steps into a Script

Insert Think Time Steps

The time that a user waits between performing successive actions is known as the *think time*. Vusers use the **lr_think_time** function to emulate real-user think time. When you record a Vuser script, VuGen records the actual think times and inserts appropriate **lr_think_time** statements into the Vuser script. You can edit the recorded **lr_think_time** statements, and manually add more **lr_think_time** statements to a Vuser script.

To insert a think time step:

Select **Design > Insert in Script > New Step > Think Time** and specify the desired think time - in seconds.

Note: When you record a Java Vuser script, **lr_think_time** statements are not generated in the Vuser script.

You can use the runtime settings to influence how the **lr_think_time** statements operate when you execute a Vuser script. For details, see the **General > Think Time** view in the runtime settings.

Insert Debug Messages

You can add a debug or error message using VuGen's user interface. For debug messages you can indicate the level of the text message—the message is only issued when your specified level matches the message class. You set the message class using **lr_set_debug_message**.

To insert a debug message:

Select **Design > Insert in Script > New Step**. Double-click **lr_debug_message** in the Steps Toolbox.

Insert Error and Output Messages

For protocols that support the **Step Navigator**, such as Web, Winsock, and Oracle NCA, you can add an error or output message using the user interface. A common usage of this function is to insert a conditional statement, and issue a message if the error condition is detected.

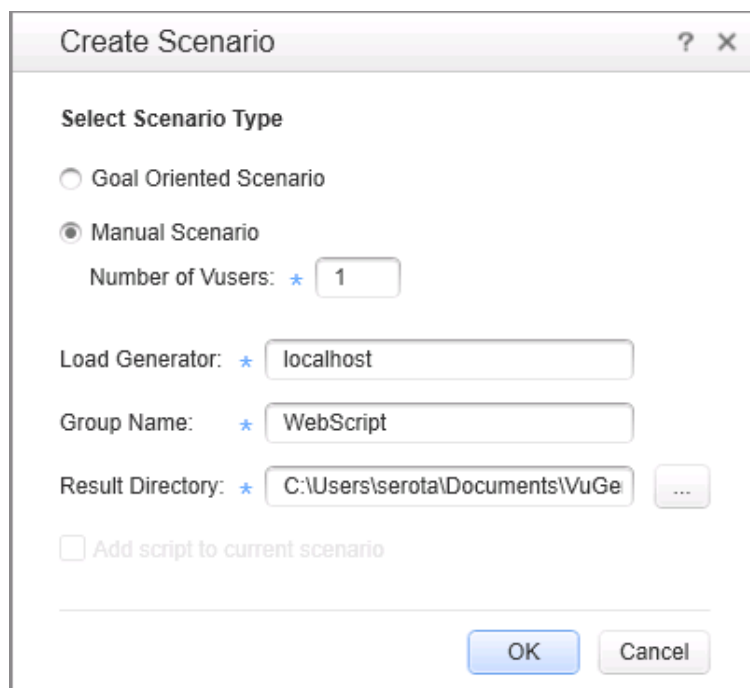
To insert an error or output message:

Select **Design > Insert in Script > New Step > Error Message** or **Output Message**, and enter the message. An **lr_error_message** or **lr_output_message** function is inserted at the current point in the script.

Note: An Error Message step in a script does not automatically set the Replay status to Failed.

Create Controller Scenario Dialog Box

This dialog box enables you to create a basic Controller scenario from within VuGen, if you have LoadRunner installed.



To access	Tools > Create Controller Scenario
Relevant tasks	"Create a Controller Scenario from VuGen" on the previous page

User interface elements are described below:

UI Element	Description
Add script to current scenario	If a scenario is currently open in the Controller and you want to add the script to this scenario, select this check box. If you clear the check box, LoadRunner opens a new scenario with the specified number of Vusers.
Group Name	For a manual scenario, Vusers with common traits are organized into groups. Specify a new group name for the Vusers.
Load Generator	The name of the machine that will run the scenario.
Results Directory	Enter the desired location for the results.
Script Name	For a goal-oriented scenario, specify a script name.
Select Scenario Type	<ul style="list-style-type: none"> • Goal Oriented Scenario. LoadRunner automatically builds a scenario based on the goals you specify. • Manual Scenario. The scenario is created manually by specifying the number of Vusers to run.

Parameterization

The **Parameters** section describes how to insert, define and modify parameters.

Parameterization Overview

When you record a business process, VuGen generates a script that contains the actual values used during recording. Suppose you want to perform the script's actions (query, submit, and so forth) using different values from those recorded. To do this, you replace the recorded values with parameters. This is known as *parameterizing* the script.

The resulting Vusers substitute the parameter with values from a data source that you specify. The data source can be either a file, or internally generated variables. For details, see ["Parameter Types" on page 345](#).

Delimiters

Parameters appear inside a Vuser script within parameter delimiters. By default, VuGen uses "{" and "}" as the left and right parameter delimiters, but you can modify these delimiters if required. In addition, you can modify the background color and outline color of parameters in a script. For details, see ["Scripting Options Tab" on page 96](#).

Script section as recorded.

```
"value=UNIX"
```

Script section after "UNIX" has been replaced with the "Operating System" parameter.

```
"value={Operating System}"
```

Input/Output parameters

You can parameterize only the arguments within a function. You cannot parameterize text strings that are not function arguments. In addition, not all function arguments can be parameterized. For details on which arguments you can parameterize, see the [Function Reference \(Help > Function Reference\)](#) for each function.

Input parameters are parameters whose values you define in the design stage before running the script. Output parameters you define during design stage, but they acquire values during test execution. Output parameters are often used with Web Service calls. Use care when selecting a parameter for your script during design stage, make sure that it is not an empty Output parameter.

Let's say you recorded a Vuser script while operating a Web application. VuGen generated the following statement that searches a library's database for the title "UNIX":

```
web_submit_form("db2net.exe",  
    ITEMDATA,  
    "name=library.TITLE",  
    "value=UNIX",  
    ENDITEM,  
    "name=library.AUTHOR",  
    "value=",  
    ENDITEM,  
    "name=library.SUBJECT",  
    "value=",  
    ENDITEM,  
    LAST);
```

When you replay the script using multiple Vusers and iterations, you do not want to repeatedly use the same value, UNIX. Instead, you replace the constant value with a parameter:

```
web_submit_form("db2net.exe",  
    ITEMDATA,  
    "name=library.TITLE",  
    "value={Book_Title}",
```

```
ENDITEM,  
"name=library.AUTHOR",  
"value=",  
ENDITEM,  
"name=library.SUBJECT",  
"value=",  
ENDITEM,  
LAST);
```

For task details, see ["Create Parameters" on page 347](#).

Correlation

To enable some recorded Vuser scripts to replay correctly, it may be necessary to implement correlation. Correlation is used when a recorded script includes a dynamic value (such as a session ID) and therefore cannot be successfully replayed. To resolve this, you convert the dynamic value into a variable—thereby enabling your script to replay successfully. For details, see ["Correlation Overview" on page 232](#).

VTs and Parameterization

VTs (Virtual Table Server) is a web-based application that works with Vuser scripts. VTs offers an alternative to standard VuGen parameterization.

When you use standard parameterization, each Vuser is assigned parameter values from a dedicated set of values - parameter values are not shared between Vusers. In contrast, VTs enables you to assign parameter values from a single set of parameter values to multiple Vusers. This may enable you to more accurately emulate a real-user environment.

VTs is composed of two components, VTs-Client and VTs-Server. VTs-Client is a set of API functions that are used to access data in VTs-Server. Because the VTs API functions integrate with VuGen, there is no need to install VTs-Client. VTs-Server includes a table that contains parameter values that can be used by your Vuser scripts. The VTs table is composed of columns and rows. Each column represents a set of values that can be assigned to a specific parameter in your Vuser scripts. The cells within a column contain the actual values that are assigned to the parameter.

Note: Significant changes were made to VTs in LoadRunner version 11.52. When upgrading to VTs 11.52 or later, these changes may result in various compatibility issues. For details on script modifications that are required in order to resolve these compatibility issues, see the VTs documentation that is available from the **VTs > Help** menu.

To install VTs:

Locate the setup file in the **Additional Components** folder of the installation media. After installing the VTs server, you can access further information from the **VTs > Help** menu. For details, see ["Installing the Virtual Table Server \(VTs\)" on page 876](#).

For details on how to use VTS functionality in TruClient Vuser scripts, see the [TruClient Help Center](#) (select the relevant version).

Parameter Types

Every parameter is defined by the type of data it contains. This section contains information on the different parameter types.

File Type Parameters

Data files hold data that a Vuser accesses during script execution. Data files can be local or global. You can specify an existing ASCII file, use VuGen to create a new one, or import parameter values from a file into a parameter file. Data files are useful if you have many known values for your parameter.

The data in a data file is stored in the form of a table. One file can contain values for many parameters. Each column holds the data for one parameter. Column breaks are marked by a delimiter, for example, a comma.

In the following example, the data file contains ID numbers and first names:

```
id,first_name
120,John
121,Bill
122,Tom
```

Note: When working with languages other than English, save the parameter file as a UTF-8 file. In the Parameter Properties window, click **Edit with Notepad**. In Notepad, save the file as a text file with UTF-8 type encoding.

For details on how to import parameter values from a file, see ["Import Parameter Values from a File" on page 348](#).

Table Type Parameters

The Table parameter type is meant for applications that you want to test by filling in table cell values. Whereas the file type uses one cell value for each parameter occurrence, the table type uses several rows and columns as parameter values, similar to an array of values. Using the table type, you can fill in an entire table with a single command. This is common in SAP GUI Vusers where the **sapgui_table_fill_data** function fills the table cells.

For details on how to import parameter values from a file, see ["Import Parameter Values from a File" on page 348](#).

XML Type Parameters

Used as a placeholder for multiple valued data contained in an XML structure. You can use an XML type parameter to replace the entire structure with a single parameter. For example, an XML parameter called **Address** can replace a contact name, an address, city, and postal code. Using XML parameters for this

type of data allows for cleaner input of the data, and enables cleaner parameterization of Vuser scripts. We recommend that you use XML parameters with Web Service scripts or for SOA services.

Internal Data Type Parameters

Internal data is generated automatically while a Vuser runs, such as Date/Time, Group Name, Iteration Number, Load Generator Name, Random Number, Unique Number, and Vuser ID.

- Custom: You can specify the parameter data type.
- Date/Time: The current date/time. You can specify the format and the offset in the Parameter Properties dialog box.
- Group Name: The name of the Vuser Group. If there is no Vuser Group (for example, when running a script from VuGen) the value is always **none**.
- Iteration Number: The current iteration number.
- Load Generator Name: The name of the Vuser script's load generator (the computer on which the Vuser is running).
- Random Number: A random number within a range of values that you specify.
- Unique Number: Assigns a range of numbers to be used for each Vuser. You specify the start value and the block size (the amount of unique numbers to set aside for each Vuser). For example, if you specify a start value of 1 and a block size of 100 the first Vuser can use the numbers 1 to 100, the second Vuser can use the numbers 201-300, and so on.
- Vuser ID: The ID number assigned to the Vuser by the Controller during a scenario run. When you run a script from VuGen, the Vuser ID is always -1.



Note: This is not the ID number that appears in the Vuser window—it is a unique ID number generated at runtime.

User-Defined Function Parameters

Data that is generated using a function from an external DLL. A user-defined function replaces the parameter with a value returned from a function located in an external DLL.

Before you assign a user-defined function as a parameter, you create the external library (DLL) with the function. The function should have the following format:

```
__declspec(dllexport) char *<functionName>(char *, char *)
```

The arguments sent to this function are both NULL.

When you create the library, we recommend that you use the default dynamic library path. That way, you do not have to enter a full path name for the library, but rather, just the library name. VuGen's bin folder is the default dynamic library path. You can add your library to this folder.

The following are examples of user-defined functions:

```
__declspec(dllexport) char *UF_GetVersion(char *x1, char *x2) {return "Ver2.0";}
```

```
__declspec(dllexport) char *UF_GetCurrentTime(char *x1, char *x2) {
time_t x = tunelessly); static char t[35]; strcpy(t, ctime( =;x)); t[24] = '\0';
return t;}
```

Create Parameters

A slideshow describing how to parameterize values in your script is available in the online help provided with LoadRunner.

1. Select the value you want to parameterize.

Pane	Do this
"Editor Pane" on page 54	<p>Select the value you want to parameterize, right-click, and select Replace with Parameter.</p> <p>Note:</p> <ul style="list-style-type: none"> • When creating XML parameters in script view, you must select only the inner xml, without the bounding tags. For example, to parameterize the complex data structure <code><A>Belement<C>Celement</C></code>, select the whole string, <code>Belement<C>Celement</C></code>, and replace it with a parameter. • When parameterizing Java Record Replay or Java Vuser scripts, you must parameterize complete values, not parts of a value.
"Step Navigator Pane" on page 52	<p>Right-click a step and select Show Arguments. Click the ABC icon next to the argument that you want to parameterize.</p>

2. Create a new parameter in the **Select or Create Parameter** dialog box.
For user interface details, see ["Select or Create Parameter Dialog Box" on page 359](#).
3. Add a list of required values.
 - a. From the **Select or Create Parameter** dialog box, select **Properties**.
 - b. Create a table and add entries to serve as the list of values for your parameter.
For user interface details, see ["Parameter Properties Dialog Box" on page 359](#).
4. Modify the parameter braces. (Optional)
 - a. Open the **Configure Parameter Braces** dialog box:
 - In the **Solution Explorer** pane, right-click the **Parameters** node and select **Configure Parameter Delimiters**.
 - **Design > Parameters > Configure Parameter Delimiters**
 - **Tools > Options > Parameters**
 - b. Modify the braces that surround parameters.
For user interface details, see ["Parameter Delimiters Configuration Dialog Box" on page 374](#).

Import Parameter Values from a File

VuGen enables you to import parameter values from an ASCII file. After you import the values, VuGen saves the imported data in a regular Vuser parameter file - with a .dat extension, located [by default] in the Vuser script's *data* folder. You can import parameter values from a file for both File and Table parameter types.

Note: The language of your machine's locale settings must match the language of the parameters in the parameters file, otherwise unexpected characters may appear in the user interface.

To import parameter values from a file:

1. In the ["Parameter Properties Dialog Box" on page 359](#), from the **Parameter type** list, select **File** or **Table**.
2. Click **Import Parameter**.
3. In the Import Parameter Values From File dialog box, locate and select the source file that contains the parameter values.
4. Select the delimiter that is used in the source file.
5. If the first row of data in the source file contains column names, select **Use first row as column names**.
6. Select the columns from which to import parameter values.
7. Click **Import**. The imported parameter values appear in a table in the Parameter Properties box.



See also:

- ["Parameter Types" on page 345](#)

Work with Existing Parameters

This task describes how to replace values with pre-defined parameters.

Replace a value with a parameter

You can replace a value with an pre-defined parameter. In the script-editor, right-click on the relevant value and select one of the following options:

- **Replace with Parameter** > select a <pre-defined> parameter. The list of parameters include parameters which have the same original value or parameters that have not yet been used.
- **Replace with Parameter** > select a parameter from the **Parameter List** dialog box.

Replace multiple occurrences of a value with a parameter

You can replace multiple occurrences of a value with a parameter. To do this, in the script editor replace at least one occurrence of the value with a parameter. Right-click the parameter and select **Replace more**

occurrences. Use **Search and Replace** to replace all of the values in the script with the selected parameter.

Restore the original value

You can undo a parameter and restore the original value by right-clicking the parameter in the script editor and selecting **Restore original value**.

Data Assignment Methods for File-Type Parameters

When using File type parameters, a data file contains the parameter values that are assigned to the Vusers during script execution. VuGen lets you specify the way in which you assign data from the source to the parameters. The following methods for assigning data are available:

Sequential

Assigns data to a Vuser sequentially. As a running Vuser accesses the data table, it takes the next available row of data.

If there are not enough values in the data table, VuGen returns to the first value in the table, continuing in a loop until the end of the test.

Random

Assigns a value from a random row in the data table every time a new parameter value is requested.

When you use the Controller to run a Vuser in a Scenario, you can specify a seed number for random sequencing. Each seed value represents one sequence of random values used for test execution. Whenever you use this seed value, the same sequence of values is assigned to the Vusers in the scenario. You enable this option if you discover a problem in the test execution and want to repeat the test using the same sequence of random values.

Unique

Allocates a unique block of parameter values to each Vuser in the scenario, and then sequentially assigns values to the parameter for each Vuser from within the Vuser's block of values. Ensure that there is enough data in the table for all Vusers and their iterations. If you have 20 Vusers and you want to perform 5 iterations, your table must contain at least 100 unique values.

Note: If you will be using Network Virtualization Insights to analyze scenario results, LoadRunner will create several additional Vusers, in addition to the Vusers that you defined in the scenario scheduler. Therefore, if the scenario includes a parameter that is assigned values using the unique data assignment method, make sure that the list of parameter values contains several extra values, i.e. several values more than is required by all Vusers to run all their iterations, in order to accommodate the additional Vusers.

If you run out of unique values, VuGen behaves according to the option you select in the **When out of values** field. For more information, see "[Parameter Properties Dialog Box](#)" on page 359.

Note: For LoadRunner users: If a script uses Unique file parameterization, running more than one Vuser group with that script in the same scenario may cause unexpected scenario results. For more information about Vuser groups in scenarios, see the Function Reference.

- For details on the different data assignment and update methods, see ["Data Assignment and Update Methods for File Parameters" below](#).

Data Assignment and Update Methods for File Parameters

For File type parameters, the Data Assignment method that you select, together with your choice of Update method, affect the values that the Vusers use to substitute parameters during the scenario run.

The Data Assignment method is determined by the **Select next row** field, and the Update method is determined by the **Update value on** field.

The following table summarizes the values that Vusers use depending on which Data Assignment and Update properties you selected. It also includes an example of each for a table/file that has the following values: **Kim; David; Michael; Jane; Ron; Alice; Ken; Julie; Fred**

Update Method	Data Assignment Method		
	Sequential	Random	Unique
Each iteration	<p>The Vuser takes the <i>next</i> value from the data table for each iteration.</p> <p>Example: All the Vusers use Kim in the first iteration, David in the second iteration, Michael in the third iteration, and so on.</p>	<p>The Vuser takes a <i>new random</i> value from the data table for each iteration.</p> <p>Example: The Vusers use random values from the table for each iteration.</p>	<p>The Vuser takes a value from the next unique position in the data table for each iteration.</p> <p>Example: For a test run of 3 iterations, the first Vuser takes Kim in the first iteration, David in the second, and Michael in the third. The second Vuser takes Jane, Ron, and Alice. The third Vuser, Ken, Julie, and Fred.</p>

<p>Each occurrence (Data Files only)</p>	<p>The Vuser takes the <i>next</i> value from the data table for each occurrence of the parameter, even if it is within the same iteration.</p> <p>Example: All the Vusers use Kim in the first occurrence, David in the second occurrence, Michael in the third occurrence, and so on.</p>	<p>The Vuser takes a <i>new random</i> value from the data table for each occurrence of the parameter, even if it is within the same iteration.</p> <p>Example: The Vusers use random values for each occurrence of the parameter.</p>	<p>The Vuser takes a <i>new unique</i> value from the data table for each occurrence of the parameter, even if it is within the same iteration.</p> <p>Example: The Vuser uses a unique value from the list for each occurrence of the parameter.</p>
<p>Once</p>	<p>The value assigned in the first iteration is used for all subsequent iterations for each Vuser.</p> <p>Example: All Vusers take Kim for all iterations.</p>	<p>The random value assigned in the first iteration is used for all iterations of that Vuser.</p> <p>Example: All Vusers take the first randomly assigned value for all the iterations.</p>	<p>The unique value assigned in the first iteration is used for all subsequent iterations of the Vuser.</p> <p>Example: The first Vuser takes Kim for all iterations, the second Vuser takes David for all iterations, and so on.</p>

Note: If you select the **Sequential** method and there are not enough values in the data table, VuGen returns to the first value in the table, continuing in a loop until the end of the test.

Vuser Behavior in the LoadRunner Controller

When you set up a scenario to run a parameterized script, you can instruct the Vusers how to act when there are not enough values. The following table summarizes the results of a scenario using the following parameter settings:

- Select next row = **Unique**
- Update Value on = **Each iteration**
- When out of values = **Continue with last value**

Situation	Duration	Resulting Action
-----------	----------	------------------

More iterations than values	Run until completion	When the unique values are finished, each Vuser continues with the last value, but a warning message is sent to the log indicating that the values are no longer unique.
More Vusers than values	Run indefinitely or Run for ...	Vusers take all of the unique values until they are finished. Then the test issues an error message Error: Insufficient records for param <param_name> in table to provide the Vuser with unique data. To avoid this, change the When out of values option in the Parameter properties or the Select next row method in the Parameter properties.
One of two parameters are out of values	Run indefinitely or Run for ...	The parameter that ran out of values, continues in a cyclic manner until the values of the second parameter are no longer unique.

XML Parameters

When you create a Web Service call to emulate a specific operation, the arguments in the operation may include complex structures with many values. You can use an XML type parameter to replace the entire structure with a single parameter.

You can create several value sets for the XML elements and assign a different value set for each iteration.

The XML parameter type supports complex schema types such as arrays, Choice, and <Any> elements.

When working with Web Service Input Arguments, you may encounter arrays and their sub-elements. You can define a single XML parameter that will contain values for all of the array elements.

You can create new XML type parameters directly from the Insert menu, similar to all other parameter types. For Web Services type scripts, you create an XML parameter directly from the Web Services Call properties.

Note: For protocols using XML, replay fails to create a request when a parameterized input argument contains the ampersand (&) character.

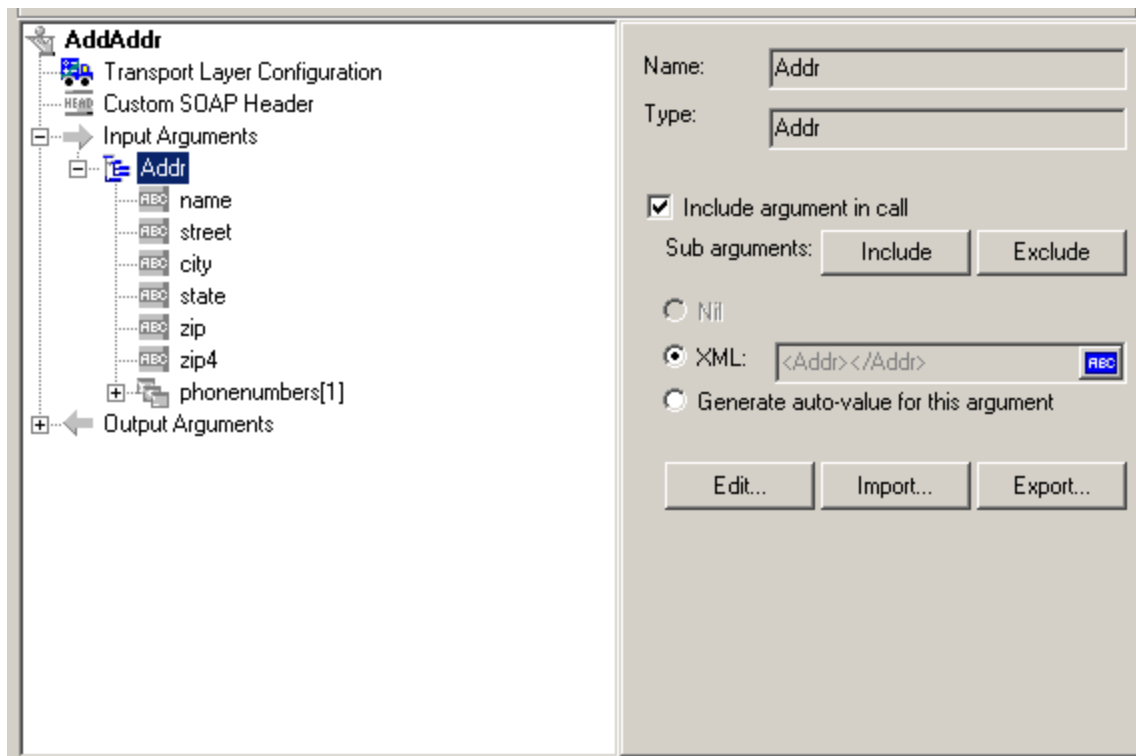
Create an XML Parameter from a Web Service Call


This task describes how to create a new XML Parameter from a Web Service Call. This procedure is in addition to the standard procedure to create a parameter. XML Parameters can also be created by using the standard procedure.

To create an XML parameter from a Web Service Call:

1. Select the root element of the complex data structure. The right pane displays the argument's

details.



2. Select **XML** in the right pane, and click the **ABC**  icon. The Select or Create Parameter dialog box opens.
3. In the **Parameter name** box, enter a name for the parameter.
4. In the **Parameter type** box, select **XML** if it is not already selected.
5. Click **Properties** to assign a value set now, or **OK** to close the dialog box and assign values later.

Create XML Parameters - Standard Method

This task describes how to create an XML type parameter without viewing the properties of a Web Service call. This is the most common way of parameterizing values for most protocols and parameter types.

For Web Service Scripts, we recommend that you create parameters from within a Web Service Call, as described in ["XML Parameters" on the previous page](#).

To create a new XML parameter:

1. Select **Insert > New Parameter** or select a constant value in the Script view and select **Replace with a Parameter** from the right-click menu. The Select or Create Parameter dialog box opens.
2. In the **Parameter name** box, enter a name for the parameter.
3. In the **Parameter type** box, select **XML** if it is not already selected.
4. Click **Properties** to assign a value set now, or **OK** to close the dialog box and assign values later.

Define XML Value Sets

This task describes how to create value sets for XML parameters.

Value sets are arrays that contain a set of values. Using the **Add Column** and **Duplicate Column** buttons, you can create multiple value sets for your parameter and use them for different iterations.

Schema	Set 1	Set 2	Set 3
Addr			
name	John Doe	Tom Smith	Kim Jones
street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
city	Delray Beach	NIL	NIL
state	FL	AZ	MA
zip	33452	NIL	02134
zip4			
phonenumber			
PhoneNumber [...]			
PhoneNumber[1]	NIL	NIL	NIL

When using value sets, the number of array elements per parameter does not have to be constant.

You can use optional elements that will appear in one value set, but not in another. This allows you to vary the values you send for each of the iterations—some iterations can include specific array elements, while other iterations exclude them.

To exclude an optional element, click the small triangle in the upper left corner of the cell and insure that it is not filled in.

In the following example, **Set 1** and **Set 2** use the optional elements: **name**, **street**, and **state**. **Set 3** does not use a street name.

Schema	Set 1	Set 2	Set 3
Addr			
name	John Doe	Tom Smith	Kim Jones
street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
city	Delray Beach	NIL	NIL
state	FL	AZ	MA
zip	33452	NIL	02134
zip4			
phonenumber			
PhoneNumber [...]			
PhoneNumber[1]	NIL	NIL	NIL

To set parameter element values

1. **View the Parameter Properties.**

If the Parameter Properties dialog box is not open, select **Vuser > Parameter List** and select the desired parameter. The dialog box shows a read-only view of the parameter values.

File Path:

Schema	Set 1	Set 2	Set 3
- Addr			
name	John Doe	Tom Smith	Kim Jones
street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
city	Delray Beach		
state	FL	AZ	MA
zipcode	33452		02134
phonenumber			
birthday			

2. Open the Data Parameterization box.

Click the **Edit Data** button to open the Data Parameterization dialog box.

Data parameterization

Schema	Set 1	Set 2	Set 3
- Addr			
name	John Doe	Tom Smith	Kim Jones
street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
city	Delray Beach		
state	FL	AZ	MA
zipcode	33452		02134
phonenumber			
birthday			

3. Define value sets for the XML parameter.

In the **Set** columns, insert values corresponding to the schema.

If a row says **NIL**, it implies that the element is nillable. To include a value for the nillable element, enter the value as usual. To mark a value as **nil**, click the NIL icon to fill it in. This erases any value

that you may have assigned to the element. In the following example, the **city** element is nillable, but it is only marked as nil in **Set 2** and **Set3**—not in **Set 1**.

Schema	Set 1	Set 2	Set 3
[-] Addr			
[ABC] name	John Doe	Tom Smith	Kim Jones
[ABC] street	2 Maple Ln.	33 Acorn Dr.	45 Jasper Ave.
[ABC] city	NIL Delray Beach	NIL	NIL
[ABC] state	FL	AZ	MA
[ABC] zip	NIL 33452	NIL	NIL 02134
[-] zip4			
[-] phonenumber			
[-] PhoneNumber [...]			
[-] PhoneNumber[1]	NIL	NIL	NIL

4. Create additional value sets.

To insert more value sets, click **Add Column** and insert another set of values in the new column. To copy an existing value set, select a row in the value set you want to copy and click **Duplicate Column**.

5. Copy arrays.

To duplicate an array element and its children, select the parent node and choose **Duplicate Array Element** from the right-click menu.

Schema	Set 1	Set 2	Set 3
[-] phone-numbers			
[-] PhoneNumber [...]			
[-] PhoneNumber[1]	[]	[]	[]
[ABC] description	Home	Home	Home
[ABC] phone-number	888-8888	111-1111	444-4444
[-] PhoneNumber[2]	[]	[]	[]
[ABC] description	Office	Office	Office
[ABC] phone-number	666-6666	222-2222	999-9999
[-] PhoneNumber[3]	[]	[]	[]
[ABC] description		bile	Mobile
[ABC] phone-number		8-3333	123-4567

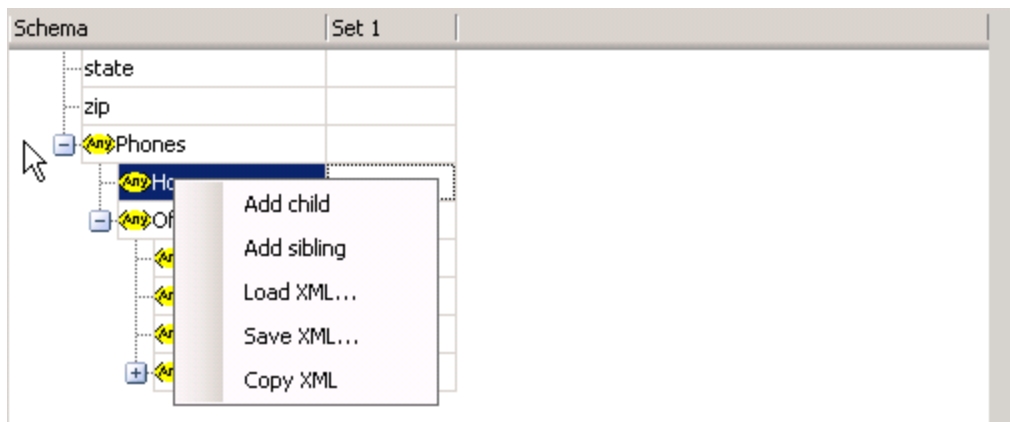
6. Handle the <any> elements.

For **any** type elements, right-click **<any>** in the **Schema** column and select one of the available options. These options may vary depending on the location of the cursor.

- **Add Array Element.** Adds a sub-element under the root element.
- **Insert child.** Adds a sub-element to the selected element.
- **Insert sibling.** Adds a sub-element on the same level as the selected element.
- **Load XML.** Loads the element values from an XML file.
- **Save XML.** Saves the array as an XML file.

- **Copy XML.** Copies the full XML of the selected element to the clipboard.

Click the **Rename** text to provide a meaningful name for each array element.



7. **Remove unwanted columns.**

To remove a value set, select it and click **Delete Column**.

8. **Save the changes.**

Click **Apply** to save the changes and update the view in the Parameter Properties dialog box.

Set an Assignment Method

This task describes how to set an assignment method. The assignment method indicates which of the value sets to use and how to use them. For example, you can instruct Vusers to use a new value set for each iteration and use the value sets sequentially or at random. For more information, see ["Data Assignment and Update Methods for File Parameters" on page 350](#).

To define an assignment method

1. Open the Parameter Properties and select a parameter.
2. Define a data assignment method.

In the **Select next value** list, select a data assignment method to instruct the Vuser how to select the file data during Vuser script execution. The options are: **Sequential**, **Random**, or **Unique**. For more information, see ["Data Assignment Methods for File-Type Parameters" on page 349](#).

3. Select an update option for the parameter.

In the **Update value on** list, select an update option. The choices are **Each Iteration**, **Each Occurrence**, and **Once**. For more information, see ["Data Assignment and Update Methods for File Parameters" on page 350](#).

4. If you chose **Unique** as the data assignment method, the **When out of values** and **Allocate Vuser values in the Controller** options become enabled.
 - **When out of values.** Specify what to do when there is no more unique data: **Abort Vuser**, **Continue in a cyclic manner**, or **Continue with last value**.


- **Allocate Vuser values in the Controller.** (for LoadRunner users only) Indicate how to allocate data blocks of parameter values to the Vusers. You can allow the Controller to automatically allocate a block size or you can specify the block size for each Vuser.

- **Automatically allocate block size.** The block size is calculated by dividing the number of parameter values by the number of Vusers.
- **Allocate x values for each Vuser.** Specify the number of values to allocate to each Vuser.

To track this occurrence, enable the **Extended Log > Parameter Substitution** option in the Log Runtime Settings. When there are not enough parameter values, VuGen writes a warning message to the Vuser log: **No more unique values for this parameter in table <table_name>.**

5. In the Parameter Properties dialog box, click **Close**.


The list of input arguments is replaced by the parameter name, and ABC button is replaced by a table

icon  which you can click to edit the parameter properties or un-parameterize the parameter.

Modify XML Parameter Properties

This task describes how to modify XML parameter properties.

To modify XML parameter properties:

1. In the Step Navigator, right-click the required step and select **Show Arguments**. The Web Service Call Properties dialog box opens.
2. From the list of arguments, under **Input Arguments**, select the XML parameter. The right pane displays the parameter details.
3. To modify the XML parameter properties, select the **XML** check box, click the table icon  button, and then select **Parameter Properties**. The Parameter Properties dialog box opens.
4. Modify the parameter properties as desired.

Set AUT Environment Parameters

When working with scripts stored in Application Lifecycle Management (ALM), you can work with different Application Under Test (AUT) environments.

AUT Environments are environments that you define in ALM that represent different testing configurations. By parameterizing the environment data, you can make your test more flexible and portable. Instead of running several scripts that use the same logic, but with different AUT environment constants, you can maintain a single script that uses AUT environment parameters.

You define environment-specific parameters in ALM's AUT Environment configuration. During the test run, ALM inserts these values into your script. For more details on working with AUT environments, refer to the *HPE Application Lifecycle Management User Guide*.

This task describes how to define an AUT environment type parameter in your Vuser script.


1. Create a parameter in VuGen. Make sure the name of the parameter matches the name of the corresponding AUT environment parameter.
2. Set the parameter type to "**Custom parameter**".
3. Enter the following parameter description "ALMPARAM" , using the exact spelling and case.

Select or Create Parameter Dialog Box

This dialog box enables you to create a new parameter or modify an existing parameter.

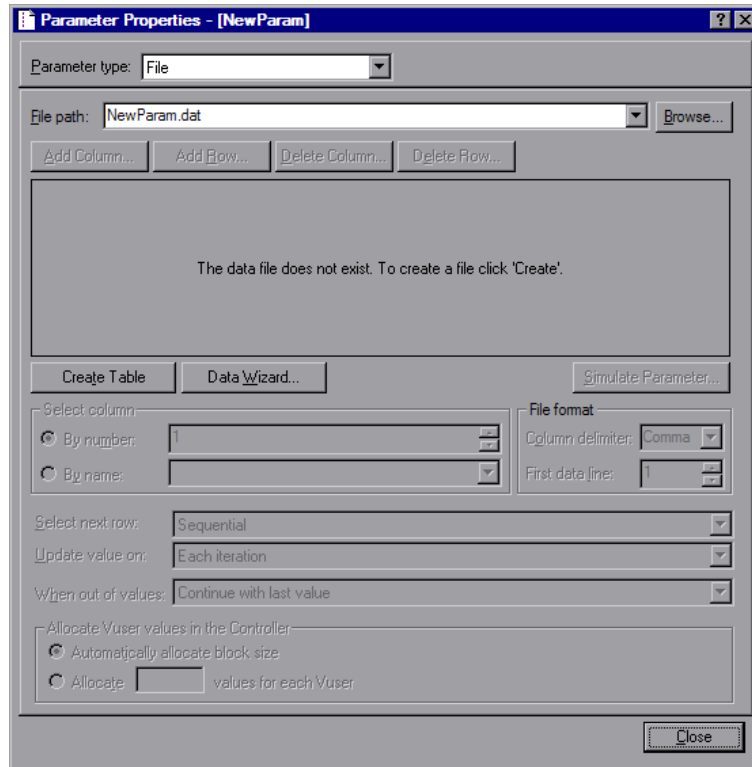
To access	Use one of the following: <ul style="list-style-type: none"> • VuGen > Solution Explorer pane > right-click on the Parameters node > Create New Parameter • In script editor, right-click on the value > Replace with Parameter > Create New Parameter • Design > Parameters > Create New Parameter
Relevant tasks	"Create Parameters" on page 347

User interface elements are described below:

UI Element	Description
Parameter name	The name of the parameter. Note: Do not use the name unique , it is used by VuGen.
Parameter type	The type of the parameter. For information about the different parameter types see "Parameter Types" on page 345 .
Original value	The original value of the parameter before parameterization.
	Opens the Parameter Properties dialog box. For details, see "Parameter Properties Dialog Box" below .

Parameter Properties Dialog Box

This page allows you to view and modify the properties of a parameter. This dialog box varies depending on the type of parameter you are using.

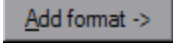
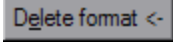
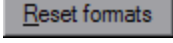


To access

VuGen > Right-click parameter > Parameter properties

Date/Time, Group Name, Iteration Number, Load Generation Name, and Vuser ID Parameters

User interface elements are described below:

UI Element	Description
	Adds the custom format specified in the Date/time format or Text format field to the format list.
	Deletes the selected format from the format list.
	Restores the format list to it's default state.
Date/time format / Text format	You can specify a custom format here. See the chart below for a list of Date/time symbols.
Format list	The list of formats. See the chart below for a list of Date/time symbols.


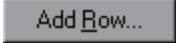

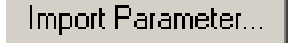

UI Element	Description
Offset (Date/time to type only)	<p>Allows you to set an offset for the date/time parameter. For example, if you want to test a date next month, you set the date offset to 30 days.</p> <ul style="list-style-type: none"> • Working days only. Use values for work days only (excludes Saturdays and Sundays). <div> <p>Note: To change the non-working days, configure the NonWorkingDays parameter under the Misc section in the vugen.ini file:</p> <pre>[Misc] NonWorkingDays="5,6"</pre> <p>Days are represented by integers as follows:</p> <p>Mon = 1, Tue = 2, Wed = 3, Thur = 4, Fri = 5, Sat = 6, Sun = 7</p> </div> <ul style="list-style-type: none"> • Prior to current date. Sets the offset for a date or time that has already passed (negative offset).
Parameter type	The parameter type. For more information see "Parameter Types" on page 345 .
Sample (current time)	Displays an example parameter value based on the selected format.
Update values on	<ul style="list-style-type: none"> • Each occurrence. Use a new value for each occurrence of the parameter in your script. This is useful when the statements using a parameter are unrelated. For example, for random data, it may be useful to use a new value for each occurrence of the parameter. • Each iteration. Updates the parameter one time per iteration. If a parameter appears in a script several times, the Vuser uses the same value for all occurrences of the parameter, for the entire iteration. This is useful when the statements using a parameter are related. <div> <p>Note: If you create an action block with parameters using its own iteration count—if you instruct VuGen to update their values each iteration, it refers to the global iteration and not the block iteration.</p> </div> <ul style="list-style-type: none"> • Once. Updates the parameter value only once during the scenario run. The Vuser uses the same parameter value for all occurrences and all iterations of the parameter. This type may be useful when working with dates and times.


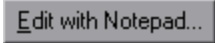
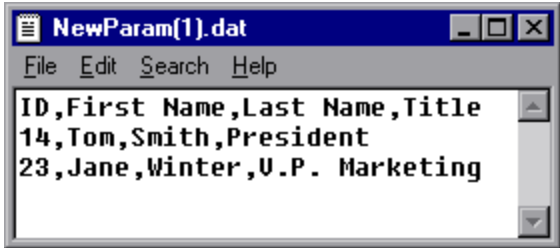

The following table describes the date/time symbols:

Symbol	Description
c	complete date and time in digits
#c	complete date as a string and time
H	hours (24 hour clock)
I	hours (12 hour clock)
M	minutes
S	seconds
p	AM or PM
d	day
m	month in digits (01-12)
b	month as a string - short format (e.g. Dec)
B	month as a string - long format (e.g. December)
y	year in short format (e.g. 03)
Y	year in long format (e.g. 2013)

File Parameters

User interface elements are described below:



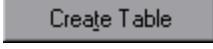
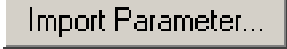


UI Element	Description
	Adds a column to the data set.
	Adds a row to the data set.
	Creates a new data table.
	Opens the Import Parameter Values from File dialog box, enabling you to import parameter values from an ASCII file. For more information, see "Import Parameter Values from a File" on page 348 .
	Deletes a column from the data set.


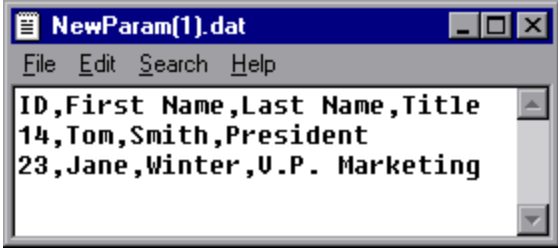
UI Element	Description
	Deletes a row from the data set.
	<p>Enables you to view and edit parameter values in Notepad. This is important when working with large data sets because VuGen displays only up to 100 rows in the UI.</p> <p>Notepad opens with the parameter's name in the first row and its original value in the second row. Enter additional column names and values into the file using a delimiter such as a comma or a tab to indicate a column break. Begin a new line for each table row (for each new row of data).</p> 
	Opens the Parameter Simulation dialog box. This allows you to simulate the parameter behavior with your data set. For more information, see "Parameter Simulation Dialog Box" on page 369 .
Select Column	Enables you to select the column to use as the data source, either by the column number or name .
File Format	<ul style="list-style-type: none"> • Column delimiter. The character used to separate values in the data file. • First data line. The first line of data to be used during Vuser script execution. The header is line 0. To begin with the first line after the header, specify 1. If there is no header, specify 0.
Select next row	The method of selecting the file data during Vuser script execution. The options are: Sequential , Random , or Unique . For more information see "Data Assignment Methods for File-Type Parameters" on page 349 .
Update value on	The method that determines when the parameter will switch to the next value. The choices are Each Iteration , Each Occurrence , and Once . For more information see "Data Assignment Methods for File-Type Parameters" on page 349 .
When out of values	Specify what to do when there is no more unique data: Abort the Vuser , Continue in a cyclic manner , or Continue with last value .

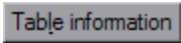
UI Element	Description
Allocate Vuser values in the Controller	<p>(LoadRunner only). Indicate how to allocate data blocks of parameter values to the Vusers. You can allow the Controller to automatically allocate the block size or you can specify the desired block size to allocate to each Vuser.</p> <ul style="list-style-type: none"> • Automatically allocate block size. The block size is calculated by dividing the number of parameter values by the number of Vusers. • Allocate x values for each Vuser. Specify the number of values to allocate to each Vuser. <p>To track this occurrence, enable the Extended Log > Parameter Substitution option in the Log Runtime Settings. When there are not enough parameter values, VuGen writes the following warning message to the Vuser log: "No more unique values for this parameter in table <table_name>".</p>
File path	Select the .dat file with the data for your parameter. Alternatively, you can create a new data set using the Create Table button.

Table Parameters

User interface elements are described below:

UI Element (A-Z)	Description
	Adds a column to the data set.
	Adds a row to the data set.
	Creates a new data table.
	Opens the Import Parameter Values from File dialog box, enabling you to import parameter values from an ASCII file. For more information, see "Import Parameter Values from a File" on page 348 .
	Deletes a column from the data set.
	Deletes a row from the data set.

UI Element (A-Z)	Description
	<p>Enables you to view and edit parameter values in Notepad. This is important when working with large data sets because VuGen only displays up to 100 rows in the UI.</p> <p>Notepad opens with the parameter's name in the first row and its original value in the second row. Enter additional column names and values into the file using a delimiter such as a comma or a tab to indicate a column break. Begin a new line for each table row (for each new row of data).</p> 
<p>Allocate Vuser values in the Controller</p>	<p>(LoadRunner only). Indicate how to allocate data blocks of parameter values to the Vusers. You can allow the Controller to automatically allocate the block size or you can specify the desired block size to allocate to each Vuser.</p> <ul style="list-style-type: none"> • Automatically allocate block size. The block size is calculated by dividing the number of parameter values by the number of Vusers. • Allocate x values for each Vuser. Specify the number of values to allocate to each Vuser. <p>To track this occurrence, enable the Extended Log > Parameter Substitution option in the Log Runtime Settings. When there are not enough parameter values, VuGen writes a warning message to the Vuser log "No more unique values for this parameter in table <table_name>".</p>
<p>Column</p>	<p>The columns to use. Alternatively, you can select Select all columns. To specify one or more columns by their number, select Columns by number and enter the column numbers separated by a comma or a dash. The column number is the index of the column containing your data. For example, if the data for the parameter is in the table's first column, select 1. In the Column delimiter box, select a column delimiter—the character used to separate the columns in the table. The available delimiters are: comma, tab, space.</p>
<p>File path</p>	<p>Select the .dat file with the data for your parameter. Alternatively, you can create a new data set using the Create Table button.</p>

UI Element (A-Z)	Description
Row delimiter for log display	This delimiter is used to differentiate between rows in the output logs. If you enable parameter substitution logging, VuGen sends the substituted values to the Replay log. The row delimiter character in the Replay log indicates a new row.
Rows	<ul style="list-style-type: none"> • Rows per iteration. How many rows to use per iteration. This only relevant when the Update value on field is set to Each iteration. If Update value on is set to Once, then the same rows will be used for all iterations. • First line of data. The first line of data to be used during script execution. To begin with the first line after the header, enter 1. •  Table information. Displays information about the table, including how many rows of data are available.
Select next row	The method of selecting the file data during Vuser script execution. The options are: Sequential , Random , or Unique . For more information see "Data Assignment Methods for File-Type Parameters" on page 349 .
Update value on	The method that determines when the parameter will switch to the next value. The choices are Each Iteration , Each Occurrence , and Once . For more information see "Data Assignment Methods for File-Type Parameters" on page 349 .
When not enough rows	Specifies what VuGen does when there are not enough rows in the table for the iteration. Example: The table you want to fill has 3 rows, but your data only has two rows. Select Parameter will get less rows than required to fill in only two rows. Select Use behavior of "Select Next Row" to loop around and get the next row according the method specified in the Select Next Row box.
When out of values	Specify what to do when there is no more unique data: Abort the Vuser , Continue in a cyclic manner , or Continue with last value .

Random Number Parameters

User interface elements are described below:

UI Element	Description
Number format	Specifies the minimum number of digits your parameter will have. Where %01d represents one digit, %02d represents two digits, and so on.

UI Element	Description
Random range	The minimum and maximum range for the random values.
Sample value	Displays sample minimum and maximum values based on the selected Number format.
Update value on	<ul style="list-style-type: none"> • Each occurrence. Use a new value for each occurrence of the parameter in your script. This is useful when the statements using a parameter are unrelated. For example, for random data, it may be useful to use a new value for each occurrence of the parameter. • Each iteration. Updates the parameter one time per iteration. If a parameter appears in a script several times, the Vuser uses the same value for all occurrences of the parameter, for the entire iteration. This is useful when the statements using a parameter are related. <p>Note: If you create an action block with parameters using its own iteration count—if you instruct VuGen to update their values each iteration, it refers to the global iteration and not the block iteration.</p> <ul style="list-style-type: none"> • Once. Updates the parameter value only once during the scenario run. The Vuser uses the same parameter value for all occurrences and all iterations of the parameter. This type may be useful when working with dates and times.

Unique Number Parameters

Note: When scheduling a scenario in the Controller, the **When out of values** option only applies to the **Run for HH:MM:SS** option in the Schedule Builder's Duration tab. It is ignored for the **Run until completion** option.

User interface elements are described below:

UI Element	Description
Number format	Specifies the minimum number of digits your parameter will have. Where %01d represents one digit, %02d represents two digits, and so on.
Number range	<ul style="list-style-type: none"> • Start. The starting value. • Block size per Vuser. The amount of unique numbers assigned to each Vuser. For example, if you specify a starting value of 1 and a block size of 100, the values be 1-100 can be used by the first Vuser, the values 101-200 can be used by the second Vuser, and so on.

UI Element	Description
Sample value	Displays an example parameter value based on the selected format.
Update value on	<ul style="list-style-type: none"> • Each occurrence. Use a new value for each occurrence of the parameter in your script. This is useful when the statements using a parameter are unrelated. For example, for random data, it may be useful to use a new value for each occurrence of the parameter. • Each iteration. Updates the parameter one time per iteration. If a parameter appears in a script several times, the Vuser uses the same value for all occurrences of the parameter, for the entire iteration. This is useful when the statements using a parameter are related. <div> <p>Note: If you create an action block with parameters using its own iteration count—if you instruct VuGen to update their values each iteration, it refers to the global iteration and not the block iteration.</p> </div> <ul style="list-style-type: none"> • Once. Updates the parameter value only once during the scenario run. The Vuser uses the same parameter value for all occurrences and all iterations of the parameter. This type may be useful when working with dates and times.
When out of values	<p>Determines what to do when the range of values is reached for a Vuser. The range of values is determined by the start value and the block size.</p> <p>Abort Vuser. Terminates the Vuser script.</p> <p>Continue in a cyclical manner. Restart the unique numbers for this Vuser from the beginning of its assigned range. For example, if a Vuser had the range of 1-100 and it reached 100, it would start again at 1.</p> <p>Continue with last value. Use the last assigned value for this parameter for all subsequent occurrences of this parameter. For example, if a Vuser had the range of 1-100 and it reached 100, it would continue with the value of 100 until the end of the script.</p>

User Defined Function Parameters

User interface elements are described below:

UI Element	Description
Function Name	The name of the function. Use the name of the function as it appears in the DLL file.
Library Names	The location of the relevant library files.

UI Element	Description
Update value on	<ul style="list-style-type: none"> • Each occurrence. Use a new value for each occurrence of the parameter in your script. This is useful when the statements using a parameter are unrelated. For example, for random data, it may be useful to use a new value for each occurrence of the parameter. • Each iteration. Updates the parameter one time per iteration. If a parameter appears in a script several times, the Vuser uses the same value for all occurrences of the parameter, for the entire iteration. This is useful when the statements using a parameter are related. <div style="background-color: #e6f2e6; padding: 10px; margin: 10px 0;"> <p>Note: If you create an action block with parameters using its own iteration count—if you instruct VuGen to update their values each iteration, it refers to the global iteration and not the block iteration.</p> </div> <ul style="list-style-type: none"> • Once. Updates the parameter value only once during the scenario run. The Vuser uses the same parameter value for all occurrences and all iterations of the parameter. This type may be useful when working with dates and times.

XML Parameters

For information about Web Services XML parameters, see ["XML Parameters" on page 352](#).


Parameter Simulation Dialog Box

This dialog box allows you to view a simulation of the behavior of a file parameter.

To access	VuGen > Parameter List > Select Parameter > Simulate Parameter
------------------	--

Important information	<ul style="list-style-type: none"> • This feature is only relevant for file type parameters. • Not all types of Parameter Substitution can be simulated. If you select Select next row: Same line as... or Update value on: Each occurrence, then the Parameter Simulation dialog box will not open. • VuGen can simulate up to 256 iterations and 256 Vusers. • Maximum parameter value length: 100 characters. • Run Indefinitely is compliant with the Real-life schedule in the Scheduler of the Controller. • If you select Select next row: Unique in the Parameter List dialog, then each Vuser is assigned a unique range of rows from which the Simulator will substitute values (for that Vuser). <div data-bbox="435 699 1412 823" style="background-color: #e6f2e6; padding: 10px; margin: 10px 0;"> <p>Note: If you have more than one unique parameter, you need to verify that each parameter has defined values for all Vusers.</p> </div> <p>With this setting, the default selection in the Allocate Vuser values in the Controller section is Automatically allocate block size. In this case, when you run the simulation, the range allocation takes place in accordance with your Scenario run mode selection. If you change the default selection to Allocate x values for each Vuser, then the Vusers will be allocated the amount of values you specify, ignoring of your Scenario run mode selection.</p>
------------------------------	--

User interface elements are described below:

UI Element	Description
Vusers	The number of Vusers to run in the simulation.
Scenario Run Mode	<ul style="list-style-type: none"> • Run until completion. Enter the number of iteration to run or select Take number of iteration from Runtime Settings. • Run indefinitely. Simulates the run indefinitely option in the controller. VuGen only actually simulates the number of iterations you specify.
	Runs the parameter simulation. The values of each parameter substitution are displayed.

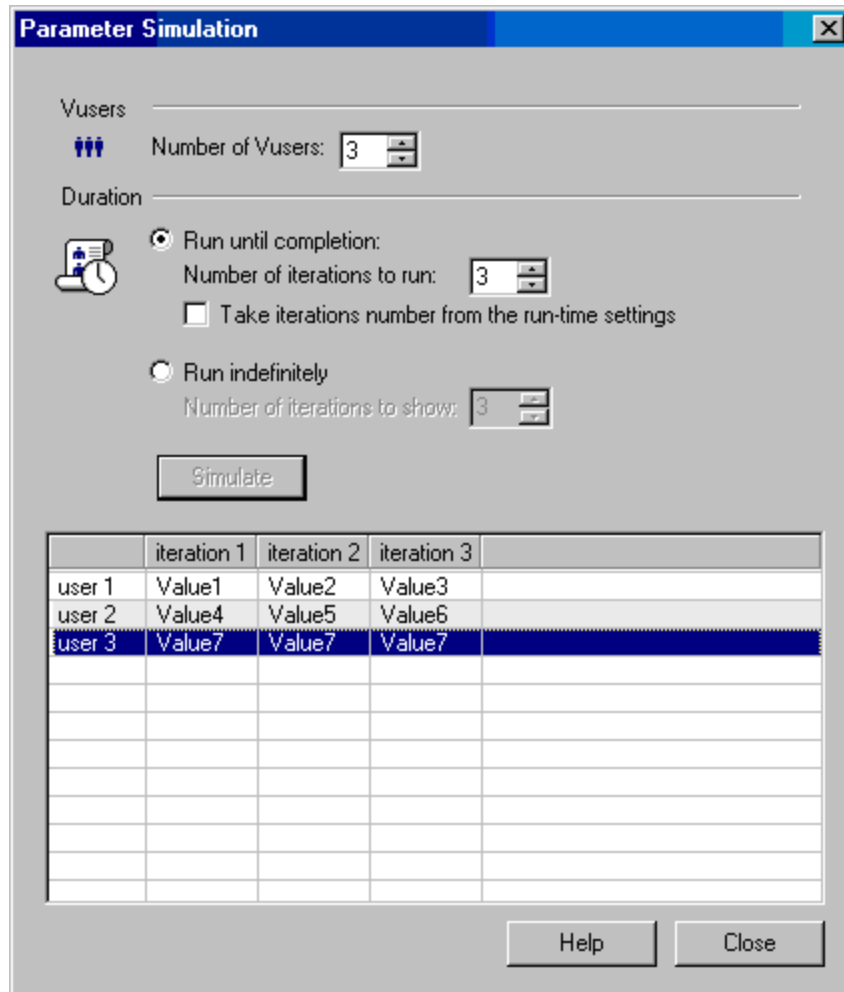
Example:

In the following examples, the settings in the Parameter List dialog box are:

- **Values for the new parameter.** Value1 to Value7
- **Select next row.** Unique
- **When out of rows.** Continue with last value
- **Allocate Vuser values in the Controller.** Automatically allocate block size

Scenario run mode: Run until completion

In the following example, the user has selected three Vusers, set the Scenario run mode to **Run until completion**, and selected three iterations.



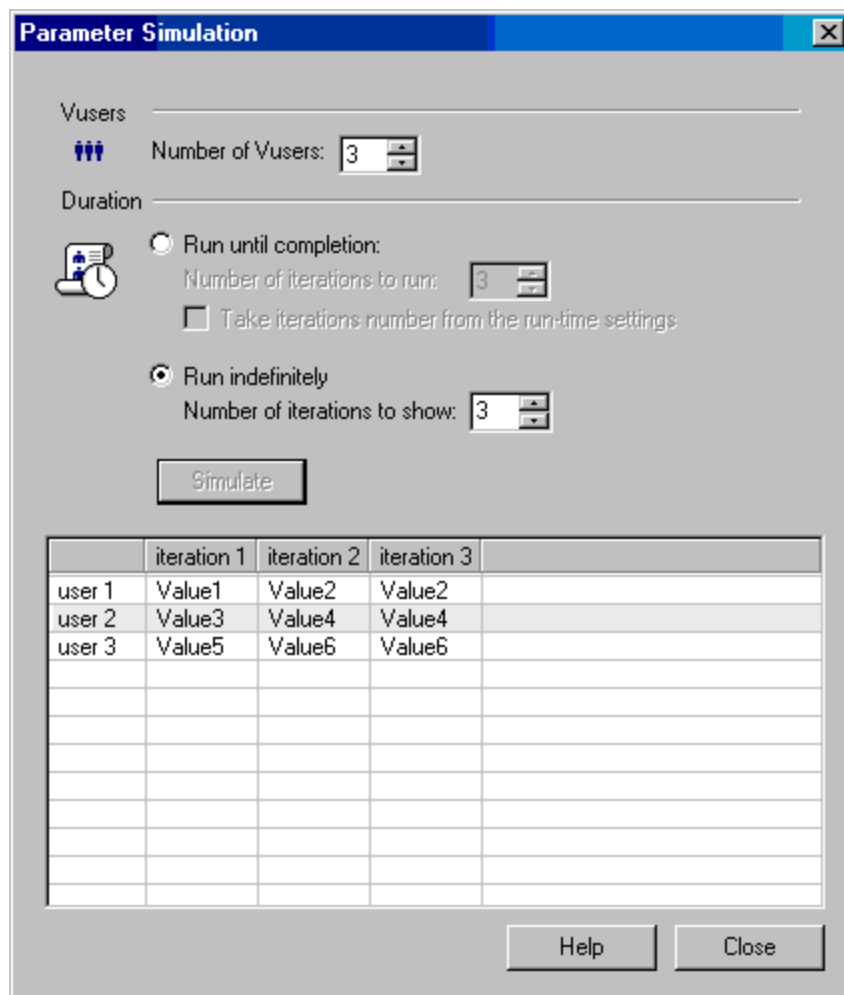
When the scenario run mode is set to **Run until completion**, the number of rows that each Vuser receives is the same as the number of iterations. The range allocation stops when there are no longer enough rows in the table.

As the simulation is run, the first Vuser takes the first three values (because this was the number of iterations). The second Vuser takes the next three values. The third Vuser takes the remaining value in the first iteration. For the remaining iterations, since the **When out of values** option in the Parameter List dialog box was set to **Continue with last value**, the third Vuser continues with the same value.

A fourth Vuser would have failed.

Scenario run mode: Run indefinitely

In the following example, the user has selected 3 Vusers and set the Scenario run mode to Run indefinitely and selected to show 3 iterations.



The dialog box is titled "Parameter Simulation". It contains the following elements:

- Vusers:** A label with a Vuser icon and a text field for "Number of Vusers" set to 3.
- Duration:** A label with a clock icon and two radio button options:
 - ☐ Run until completion:
 - Number of iterations to run: 3
 - ☐ Take iterations number from the run-time settings
 - ☒ Run indefinitely
 - Number of iterations to show: 3
- A "Simulate" button.
- A table with 4 columns: "user", "iteration 1", "iteration 2", "iteration 3", and an empty fifth column. The first three rows are populated with values for three users.
- "Help" and "Close" buttons at the bottom right.

	iteration 1	iteration 2	iteration 3	
user 1	Value1	Value2	Value2	
user 2	Value3	Value4	Value4	
user 3	Value5	Value6	Value6	

When the Scenario run mode is set to Run indefinitely, the allocated range for each Vuser is calculated by dividing the number of cells in the .dat file by the number of Vusers. In this scenario, that is $7/3 = 2$ (The simulator takes the closest smaller integer.).

As the simulation is run, the first Vuser takes Value1 and Value2. The second Vuser takes Value3 and Value4 and the third Vuser takes Value5 and Value6. Since there were only 3 Vusers, Value7 was not distributed.

Note: If you hold the mouse over the cells in the first column of the table, a tool tip appears with information about which values were assigned to that Vuser.

If you hold the mouse over cells which were not assigned values, a tool tip appears with the reason no values were assigned.




A tool tip does not appear if a proper value was assigned.

Parameter List Dialog Box

This dialog box enables you to view, create, delete, select, and modify parameters. The Parameter list shows all of the parameters that you created, including both input and output parameters.

To access	Use one of the following: <ul style="list-style-type: none"> • VuGen > Solution Explorer pane > Parameters node > Parameter List • In the script editor, right-click on a value > Replace with Parameter > Parameter List • Design > Parameters > Parameter List
Important information	Do not name a parameter <i>unique</i> , since this name is used by VuGen.
See also	"Parameter Properties Dialog Box" on page 359

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Creates a new parameter. This does not replace any highlighted text with the parameter. Do not name a parameter <i>unique</i> , since this name is used by VuGen.
	Deletes the selected parameter. <div>  Note: If the parameter replaced a previous value, the original value is restored. </div>
<Parameter Properties Pane>	This pane appears different depending on the type of parameter you are using. For information about this pane, see the relevant section in the "Parameter Properties Dialog Box" on page 359 .
Parameter type	This drop-down list lets you select the parameter type. For information about the different parameter types, see "Parameter Types" on page 345 .

Create Parameter Dialog Box

This dialog box enables you to create parameters directly from the Snapshot view.

To access	In VuGen: <ol style="list-style-type: none"> 1. After recording a script, show the Snapshot pane: View > Snapshot. 2. Click the Recording button to show the Recording snapshot. 3. Select the string you want to parameterize. 4. Select Create Parameter from the right-click menu.
------------------	---

Important Information	This dialog box is only available for protocols with snapshots, such as Web HTTP/HTML.
------------------------------	--

User interface elements are described below:

UI Element	Description
Parameter Name	The name of the parameter.
Selected Value	The string that will be substituted by a parameter.
Left Boundary	The left boundary of the string to define as a parameter.
Right Boundary	The right boundary of the string to define as a parameter.

Parameter Original Value Dialog Box

This dialog box appears when you are parameterizing a Vuser script, and lets you specify the parameter's original value.

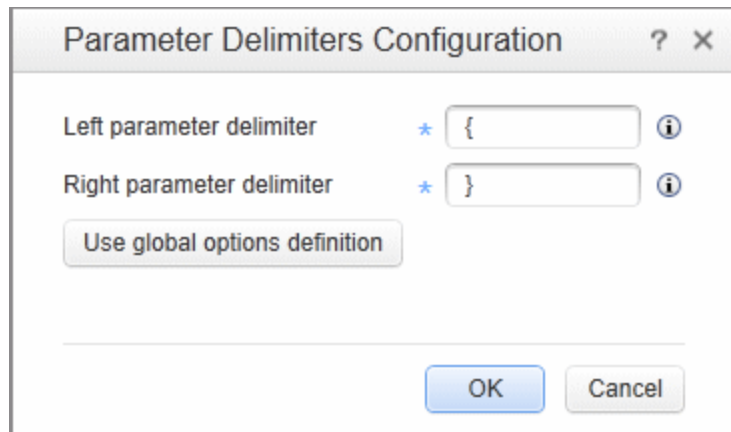
To access	In VuGen, select a text string in the Editor. Right-click and then select Replace with Parameter > Parameter List . Select an existing parameter and click Close .
Important information	Each parameter has an original value. You can replace any parameter in a Vuser script with the parameter's original value. When you parameterize a selected text string and the selected text string is not the same as the selected parameter's original value, you can select to either keep the parameter's existing original value, or replace the parameter's original value with the selected text string.

User interface elements are described below:

UI Element	Description
Use old original value <text>	Keeps the parameter's existing original value.
Use new original value <text>	Assigns the selected text as the parameter's new original value.

Parameter Delimiters Configuration Dialog Box

This dialog box enables you to define the delimiter type used to enclose the selected parameter.



To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> • VuGen > Solution Explorer pane > right-click on the Parameters node > Configure Parameter Braces • Design > Parameters > Configure Parameter Braces
Important information	<p>You can enter any character to be used as the left and right delimiters. The characters do not have to be identical. After you have defined the characters, subsequent scripts use the characters defined for right and left delimiter.</p>

User interface elements are described below:

UI Element	Description
Left parameter brace	<p>The delimiter used at the left of the parameter.</p> <p>Note: The left parameter brace does not have to be the same as the right parameter brace.</p>
Right parameter brace	<p>The delimiter used at the right of the parameter.</p> <p>Note: The left parameter brace does not have to be the same as the right parameter brace.</p>
Use global options definition	<p>This option reverts the selected delimiter to {.</p>

Troubleshooting and Limitations for Parameterization

This section describes troubleshooting and limitations for parameters.

Function Argument Limitations

You can use parameterization only for the arguments within a function. You cannot parameterize text strings that are not function arguments. In addition, not all function arguments can be parameterized. For details on which arguments you can parameterize, see the [Function Reference \(Help > Function Reference\)](#) for each function.

For example, consider the **lrd_stmt** function. The function has the following syntax:

```
lrd_stmt (LRD_CURSOR FAR *mptCursor, char FAR *mpcText, long mliTextLen, LRDOS_INT4  
mjOpt1, LRDOS_INT4 mjOpt2, int miDBErrorSeverity);
```

The indicates that you can parameterize only the *mpcText* argument.

A recorded **lrd_stmt** function could look like this:

```
lrd_stmt(Csr4, "select name from sysobjects where name =\"Kim\" ", -1, 148, -99999,  
0);
```

You could parameterize the recorded function to look like this:

```
lrd_stmt(Csr4, "select name from sysobjects where name =\"<name>\" ", -1, 148, -  
99999, 0);
```

Note: You can use the **lr_eval_string** function to "parameterize" a function argument that you cannot parameterize by using standard parameterization. In addition, you can use the **lr_eval_string** function to "parameterize" any string in a Vuser script.

For the COM and Microsoft .NET protocols, use only the **lr.eval_string** function to define a parameter. For example,
`lr.eval_string("{Custom_param}")`.

For more information on the **lr_eval_string** function, see the Function Reference.

Data Table File Size Limitations

If a .dat file's size is over 100MB, a message is displayed that the file is too big and will not be displayed.

If you need to load a file over 100MB, you can change "MaxParametersDisplaySize" parameter in vugen.ini:

```
[ParamTable]  
MaxParametersDisplaySize=100000000
```

Asynchronous Communication


Synchronous and Asynchronous Concepts

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

Originally, web applications communicated using conversations that had a synchronous nature. A typical synchronous conversation includes the following steps:

1. The user interacts with an application that is presented in a web browser.
2. Based on the user input, the browser submits a request to the web server.
3. The server sends a response to the request, and the application in the browser is updated.

Synchronous applications have a number of limitations. One limitation involves the updating of the data that is displayed in the application inside the browser. For example, consider an application that displays stock prices of a number of shares. Ideally, the application should be able to update the display of the stock prices as soon as the prices are updated on the web-server. A synchronous application would be able to update the prices on a fixed time interval.

 Every 10 seconds, the browser could sent a request to the server for the most up-to-date stock prices.

One limitation of this solution is that the displayed stock prices may be out-of-date for a period of time before the refresh interval is reached. Although this may not be critical in our share portfolio scenario, the scenario illustrates the limitation of a synchronous application to timeously update information.

Where necessary, synchronous applications are being replaced with what are known as *asynchronous* applications. Asynchronous applications enable a client to be notified whenever an event occurs on the server side. Asynchronous applications are therefore better able to update information as required.

To enable asynchronous behavior, asynchronous communication occurs in parallel (simultaneously) with the main, synchronous flow of the business processes. This behavior makes asynchronous applications harder to accurately emulate using traditional synchronous Vuser scripts.

Although there are numerous types of asynchronous applications, there are three primary types: *push*, *poll*, and *long-poll*. For details, see ["Types of Asynchronous Communication" on the next page](#).



See also:

- For an introduction to using asynchronous communication in Vuser scripts, see ["VuGen Support for Asynchronous Communication" on page 380](#).

Types of Asynchronous Communication

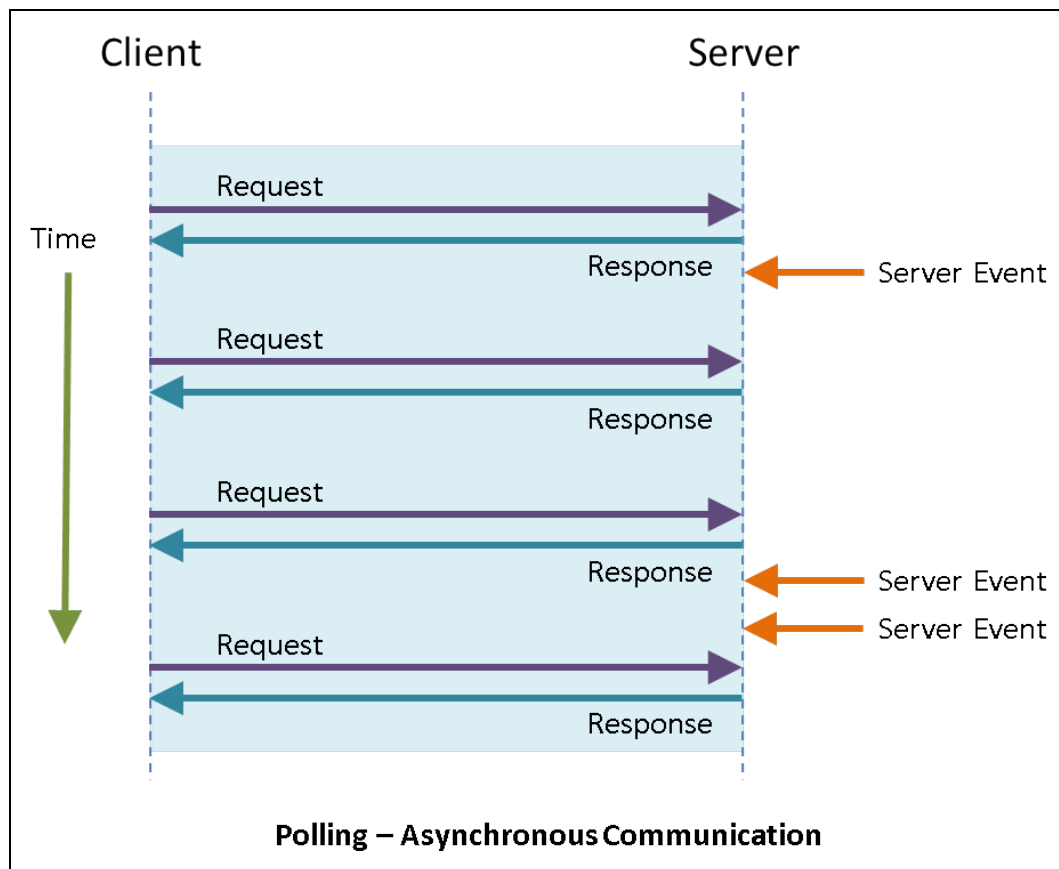
For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

Asynchronous request and response sequences

Asynchronous communication is comprised of various request and response sequences. Such request and response sequences can be classified as one of three types of asynchronous communication: *poll*, *push*, and *long-poll*. When you develop a Vuser script, it is often useful to know the types of asynchronous communication that are implemented when the required business processes are performed.

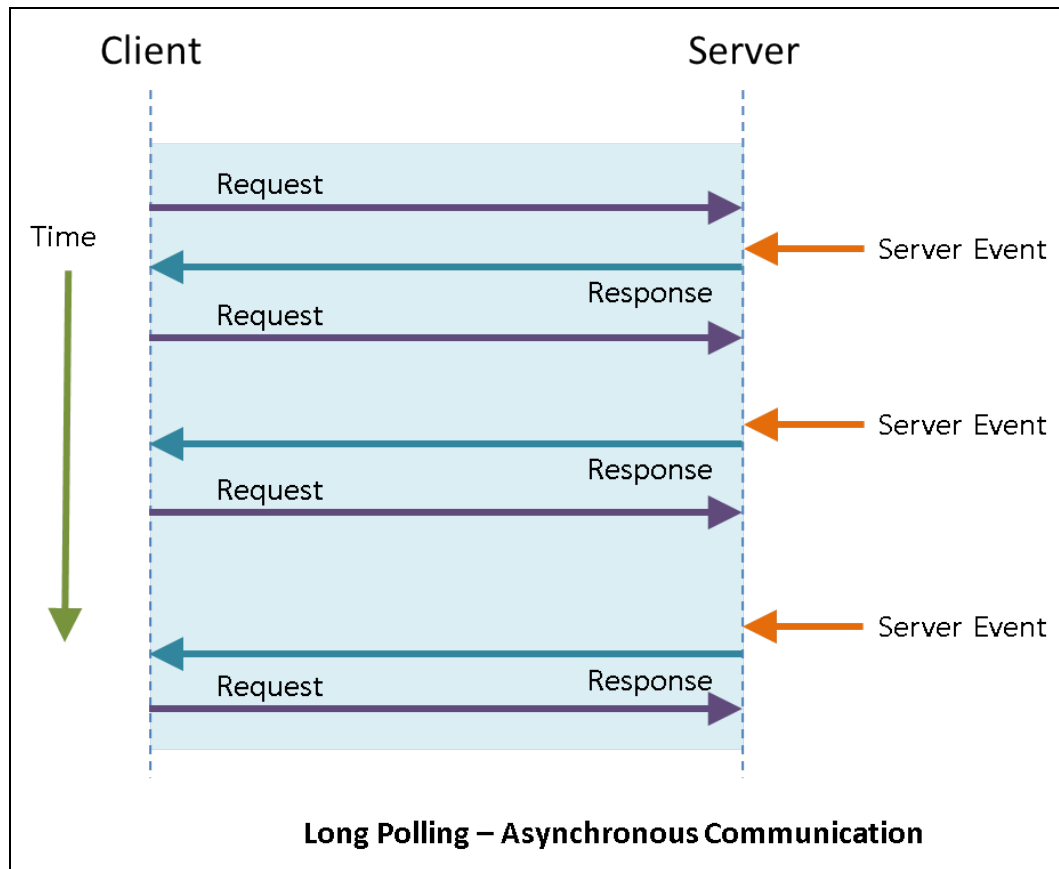
Polling Asynchronous Communication

The browser sends HTTP requests to the server at regular intervals, for example, every 5 seconds. The server responds with updates. This enables the system to intermittently update the application interface inside the browser. If the server has no update, it informs the application that there is no update, based on the application protocol.



Long-Polling Asynchronous Communication

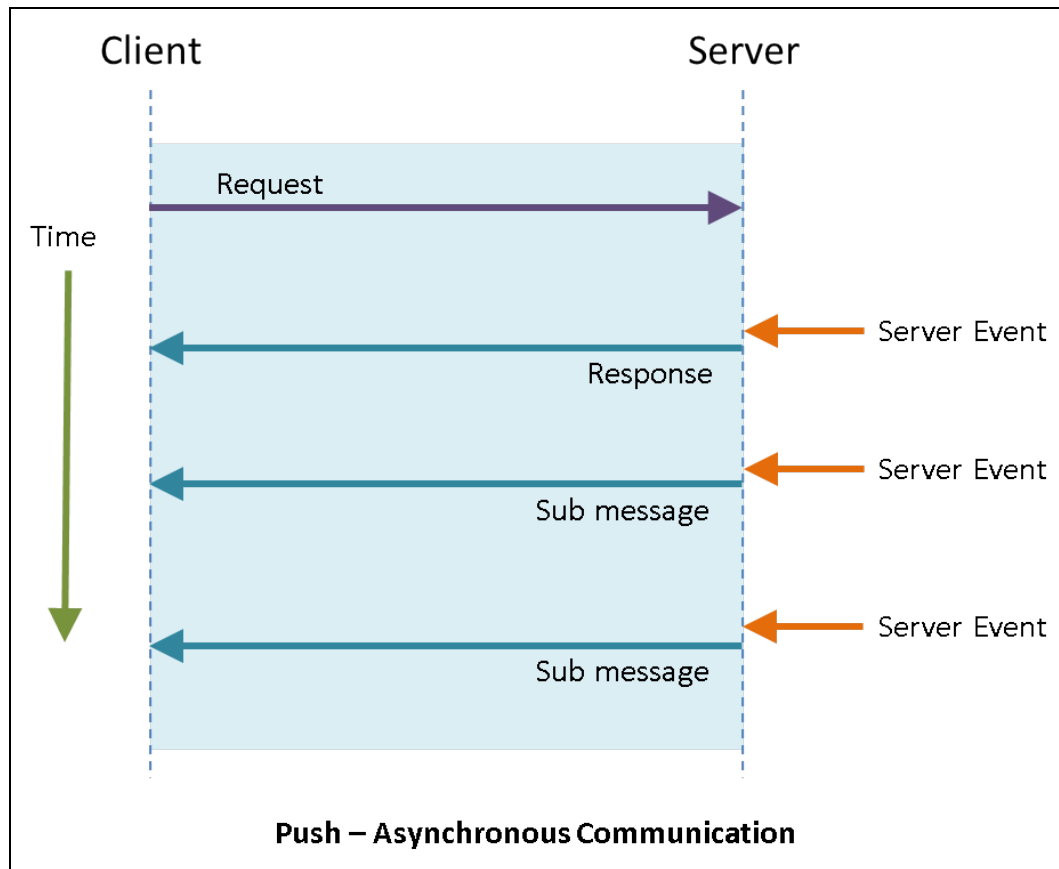
The client generates an HTTP request to a known address on the server. Whenever the server has an update, it responds with an HTTP response. Immediately after receiving the server response, the client issues another request.



Push Asynchronous Communication

The client opens a connection with the server by sending a single HTTP request to a known address on the server. The server then sends a response that appears to never end, so that the client never closes the connection. Whenever necessary, the server sends a “sub message” update to the client over the open connection. The server may or may not terminate this connection. During the time the connection is open, if the server has no real update to send, it sends "ping" messages to the client to prevent the client from closing the connection for timeout reasons.

Note: Push-type conversations are supported for Web - HTTP/HTML protocol actions inside Web - HTTP/HTML, Flex, and Web Services Vuser scripts, NOT for **Flex_amf_call** steps in Flex Vuser scripts.



VuGen Support for Asynchronous Communication

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

Web-based applications can exhibit synchronous behavior, asynchronous behavior, or a combination of both. For an introduction to these behavior types, see ["Synchronous and Asynchronous Concepts" on page 377](#). VuGen enables you to build and run Vuser scripts that emulate user activity for both synchronous and asynchronous applications. To build a Vuser script for a synchronous application, follow the typical Vuser script building process. However, to build a script for an asynchronous application, you must perform some additional tasks - beyond the typical Vuser script building process. If you record and generate a script for an application that performs asynchronous behavior - without performing the additional asynchronous-related tasks, the script may not run successfully.

Building a Vuser script for an asynchronous application begins with recording the business processes that produce the asynchronous communication. After the business processes are recorded and the Vuser script is generated, VuGen scans the generated Vuser script and attempts to locate the asynchronous communication. If asynchronous communication is detected, VuGen modifies the script - inserting the appropriate asynchronous API functions.

To enable VuGen to successfully identify the asynchronous behavior in a Vuser script, the asynchronous communication must contain at least the required minimum number of request and response sequences.

- To identify a poll-type conversation, the recorded Vuser script must contain at least three sequences with matching URLs and similar polling intervals.
- To identify a long-poll-type conversation, the recorded Vuser script must contain at least three sequences with matching URLs.

In addition, the **Async Scan** recording option must be selected. For details, see ["Create an Asynchronous Vuser Script"](#) below.

In some scenarios, the modifications that VuGen makes to the Vuser script are sufficient to enable the script to run and emulate the required asynchronous behavior. In other scenarios, additional "manual" modifications are required. For details, see ["How VuGen Modifies a Vuser Script for Asynchronous Communication"](#) on page 385.

Note: The modifications that must be made to a generated Vuser script to enable the script to emulate asynchronous behavior are dependent on the type of the asynchronous behavior: *push*, *polling*, or *long-polling*. For details, see ["Types of Asynchronous Communication"](#) on page 378.

Asynchronous communication in a Vuser script is divided into one or more conversations. The individual asynchronous conversations that VuGen detects in a Vuser script are listed in the **Async** tab of the **Design Studio**. Use this list of asynchronous conversations to systematically analyze the modifications that VuGen made to the Vuser script to make sure that VuGen has correctly identified the asynchronous behavior, and correctly modified the Vuser script to emulate the required asynchronous behavior. For details on the **Async** tab of the **Design Studio**, see ["Async Tab \[Design Studio\]"](#) on page 404.

After modifying a Vuser script to enable it to emulate asynchronous communication, it may be necessary to perform correlation activities on the modified script. For details about correlation, see ["Correlating Asynchronous Vuser Scripts"](#) on page 392.

Note: Async functionality is not supported when you replay a Vuser script in WinINet mode.

For details on how to build a Vuser script for an application that utilizes asynchronous communication, see ["Create an Asynchronous Vuser Script"](#) below.

Create an Asynchronous Vuser Script

Note:

- For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording"](#) on page 422.
- Asynchronous communication is not supported for replaying a Vuser script in WinINet mode.

Create a new Vuser script

1. Click the **New Script** button on the VuGen toolbar.
2. Select **Web - HTTP/HTML**, or one of the other Vuser protocols that support asynchronous communication.
3. Click **Create**. VuGen creates a basic Vuser script.

Enable Async Scan

1. Select **Record > Recording Options**.
2. Under **General**, select **Code Generation**.
3. Make sure that the **Async Scan** check box is selected. This instructs VuGen to scan the Vuser script after recording, locate asynchronous communication, and insert the appropriate asynchronous functionality.

Record the business processes using the typical VuGen recording process

1. Click **Record** on the VuGen toolbar.
2. Enter the required information in the Start Recording dialog box, and then click **Start Recording**.
3. Perform the business processes that the Vuser will emulate, and then click **Stop Recording** on the floating toolbar.

Note: In order for VuGen to be able to successfully identify the asynchronous behavior in a Vuser script, the asynchronous communication must contain at least the required minimum number of client requests and server responses. For details, see ["Types of Asynchronous Communication" on page 378](#).

Generate, scan, and modify the Vuser script

1. After you click **Stop Recording**, VuGen generates the Vuser script.
2. After generating the script, VuGen scans the generated script to locate instances of asynchronous communication.
3. If VuGen locates any instances of asynchronous communication, VuGen will modify the script to enable the script to run and emulate the asynchronous behavior. For details, see ["How VuGen Modifies a Vuser Script for Asynchronous Communication" on page 385](#).
4. The **Design Studio** opens. Click the **Async** tab. The **Async** tab displays a list of all instances of asynchronous communication that VuGen located in the Vuser script.

Review the modifications that VuGen made to the script

For each asynchronous conversation that appears in the **Async** tab of the **Design Studio**, perform the following tasks:

1. Open the Vuser script in the Editor.
2. Locate the **web_reg_async_attributes** step that starts the asynchronous conversation. Ensure that the **web_reg_async_attributes** step is located at the start of the asynchronous conversation.
3. Make sure that the URL parameter in the **web_reg_async_attributes** step is the same as one of the URLs that are specified in the action step that follows the **web_reg_async_attributes** step.
For details on the **web_reg_async_attributes** step, see ["Defining the Start of an Asynchronous Conversation" on page 388](#).
4. Notice that the step comment before the **web_reg_async_attributes** step contains a TODO token. The TODO token indicates that you should check the relevant callback implementations in the AsyncCallbacks.c extra file.
5. Locate the **web_stop_async** step that ends the asynchronous conversation. Ensure that the **web_stop_async** step is located at the end of the asynchronous conversation.
6. Make sure that the **web_stop_async** step runs as required. For details, see ["Fine-Tuning the End of an Asynchronous Conversation" on page 391](#).
For details on the **web_stop_async** step, see ["Defining the End of an Asynchronous Conversation" on page 389](#).
7. Review the callback implementation and make modifications to the script as required. For details, see ["Implementing Callbacks" on page 392](#).
8. Make sure that all *counter* and *complex string* parameters are set correctly. Notice that for each such parameter, a TODO comment exists and has a matching task in the **Tasks** pane. For details, see ["Parsing URLs" on page 399](#).
9. Check the **Tasks** pane for specific actions that are required in order to complete the script development process. Such actions may include verifying callback implementations, or verifying the implementation of specific parameters.
10. Once all parameters are initialized correctly, run the script to make sure that the asynchronous conversation runs as expected.

Asynchronous Communication API

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

The Function Reference (**Help > Function Reference**) includes several functions that enable Vuser scripts to emulate asynchronous communication. These asynchronous communication functions are:

web_reg_async_attributes

This function registers the next action function as the beginning of an asynchronous conversation, and defines the behavior of the asynchronous communication.

web_stop_async

This function cancels the specified asynchronous conversation, including all its active and future tasks.

web_sync

This function suspends the Vuser script execution until the specified parameter is defined.

web_util_set_request_url

This function sets the specified string to be the request URL for the next request sent in the conversation. This is applicable only when called from a callback.

web_util_set_request_body

This function sets the specified string to be the request body for the next request sent in the conversation. This is applicable only when called from a callback.

web_util_set_formatted_request_body

This function is similar to the **web_util_set_request_body** function. However, this function is included as part of a Flex protocol asynchronous conversation instead of a Web(HTTP/HTML) protocol asynchronous conversation. This function expects an XML formatted request body, which will be converted before the request is sent.

For details on the asynchronous API functions, see the **Function Reference (Help > Function Reference)**.

The **web_reg_async_attributes** function should be called before the step that starts the asynchronous conversation. The **web_reg_async_attributes** function receives a number of arguments that define the asynchronous conversation. One of these arguments is the URL of the asynchronous conversation. As soon as the replay engine downloads this URL in the step that follows the **web_reg_async_attributes** function, the asynchronous conversation begins. The callbacks that are registered in the **web_reg_async_attributes** function enable the script developer to control some of the characteristics of the asynchronous conversation (for example, to change the URL). The asynchronous conversation continues until the **web_stop_async** step, or until the end of the iteration. In a push-type conversation, the server may close the connection and thereby end the conversation.

Note: Async functionality is not supported when you replay a Vuser script in WinINET mode.

For details on how the asynchronous functions differ from synchronous functions, see "[How Asynchronous Functions Differ from Synchronous Functions](#)" below.

How Asynchronous Functions Differ from Synchronous Functions

For a list of protocols that support asynchronous communication, see "[Protocol Support for Async, IPv6, and 64-bit Recording](#)" on page 422.

The **Function Reference (Help > Function Reference)** includes several functions that enable Vuser scripts to emulate asynchronous communication. These asynchronous functions differ from the other API functions in the following ways:

- The network traffic that the asynchronous functions generate runs in parallel – simultaneously – with the main flow in the Vuser script. This means that the asynchronous communication can continue even when the synchronous steps end.
- The asynchronous communication continues even during execution of non-web functions (e.g. **lr_think_time**).
- Some of the asynchronous communication API functions use callback functions. The user needs to specify callbacks that are scheduled by the replay engine when a predefined event occurs. For details on using callbacks with asynchronous functions, see ["Implementing Callbacks" on page 392](#).

How VuGen Modifies a Vuser Script for Asynchronous Communication

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After you create a Vuser script and record the required business processes, VuGen generates the Vuser script. VuGen then scans the generated script to locate instances of asynchronous communication. This process is called an Async scan. If VuGen locates any instances of asynchronous communication in the Vuser script, VuGen modifies the script to enable the script to run and emulate the required asynchronous behavior.

Note: VuGen will scan a script for asynchronous communication only if the **Async Scan** recording option is selected. For details, see ["Create an Asynchronous Vuser Script" on page 381](#).

Asynchronous communication in a Vuser script is divided into one or more conversations. The individual asynchronous conversations that VuGen detects in a Vuser script are listed in the **Async** tab of the **Design Studio**. Use this list of asynchronous conversations to systematically analyze the modifications that VuGen made to the Vuser script during the Async scan. Make sure that VuGen has correctly identified the asynchronous behavior in the Vuser script, and correctly modified the Vuser script to emulate the required asynchronous behavior. For details on the **Async** tab of the **Design Studio**, see ["Async Tab \[Design Studio\]" on page 404](#).

Note: After modifying a Vuser script to enable it to emulate asynchronous communication, it may be necessary to perform correlation activities on the modified script. For details about correlation, see ["Correlating Asynchronous Vuser Scripts" on page 392](#).

How VuGen modifies Vuser scripts

Asynchronous behavior in a Vuser script is divided into one or more asynchronous conversations. For each asynchronous conversation, VuGen performs the following tasks:

1. VuGen inserts a **web_reg_async_attributes** step before the start of the asynchronous conversation. The **web_reg_async_attributes** step includes an ID for the asynchronous conversation. This ID is used by a subsequent **web_stop_async** step to indicate the end of the asynchronous conversation.

The Pattern argument indicates the type of the asynchronous behavior: *push*, *poll*, or *long-poll*.



```
Example: web_reg_async_attributes("Push_0",  
  
"Pattern=Push",  
  
"URL=http://push.example.com/example",  
  
"RequestCB=Push_0_RequestCB",  
  
"ResponseHeadersCB=Push_0_ResponseHeadersCB",  
  
"ResponseBodyBufferCB=Push_0_ResponseBodyBufferCB",  
  
"ResponseCB=Push_0_ResponseCB",  
  
LAST);
```

For details on how a **web_reg_async_attributes** step is used at the start of an asynchronous conversation, see ["Defining the Start of an Asynchronous Conversation" on page 388](#).

For details on the **web_reg_async_attributes** function, see the Function Reference (**Help > Function Reference**).

For details on the types of asynchronous behavior that are supported by VuGen, see ["Types of Asynchronous Communication" on page 378](#).

2. VuGen adds a comment before the inserted **web_reg_async_attributes** step. The comment includes details about the asynchronous conversation, including:
 - a. The ID of the asynchronous conversation.
 - b. The URLs that are included in the conversation.
 - c. Suggested implementations for the callback functions that are declared in the **web_reg_async_attributes** step. These implementations are added in AsyncCallbacks.c, one of the script's extra files.



Example: /* Added by Async CodeGen.

ID=Push_0

ScanType = Recording

The following URLs are considered part of this conversation:

<http://push.example.com/example>



TODO - The following callbacks have been added to AsyncCallbacks.c.

Add your code to the callback implementations as necessary.

Push_O_RequestCB

Push_O_ResponseHeadersCB

Push_O_ResponseBodyBufferCB

Push_O_ResponseCB

*/

3. For *push* conversations, VuGen inserts asynchronous API functions into the Vuser script, but does not remove any of the recorded code from the Vuser script. For *polling* and *long-polling* conversations, VuGen may remove steps or step parameters from the generated Vuser script. VuGen removes steps or step parameters in cases where the relevant URLs will be requested by running the inserted asynchronous functions - and not by running the original steps that have been removed.



Note: Removed steps are not deleted – they are commented out. You can "uncomment" these steps if required.

4. When relevant, VuGen adds a **web_stop_async** step at the end of the asynchronous conversation. The **web_stop_async** step marks the end of the asynchronous conversation. For details on the **web_stop_async** step, see the Function Reference (**Help > Function Reference**).
5. The recording snapshots are updated by grouping the tasks in the asynchronous conversation under the step that started the conversation.

How VuGen modifies flex_amf_call steps

VuGen supports asynchronous polling and long-polling behavior in **flex_amf_call** steps. Flex scripts that contain *polling* or *long-polling* in **flex_amf_call** steps are handled by VuGen just like Web(HTTP/HTML) scripts, except for the following:

- The RequestCB will contain a commented call to **web_util_set_formatted_request_body**, which can be used to pass an XML formatted request body, which will be encoded and sent with the request.
- The **aResponseBodyStr** and **aResponseBodyLen** parameters of the ResponseCB give user access to the XML representation of the response body.

Defining the Start of an Asynchronous Conversation

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After VuGen scans a Vuser script for asynchronous communication, the Async tab of the Design Studio lists the asynchronous conversations that VuGen found in the script. VuGen inserts a **web_reg_async_attributes** into the Vuser script at the start of each asynchronous conversation that was detected. Use VuGen's Step Navigator to find the associated **web_reg_async_attributes** steps in the Vuser script. The **web_reg_async_attributes** steps should be located where the asynchronous conversations start when the script runs.

A **web_reg_async_attributes** step that is added to a Vuser script includes the following parameters:

- ID
- URL
- Pattern
- PollIntervalMS (for poll-type conversations only)
- RequestCB
- ResponseBodyBufferCB (for push-type conversations only)
- ResponseHeadersCB (for push-type conversations only)
- ResponseCB

The URL parameter in the **web_reg_async_attributes** step should be the same as one of the URLs that are specified in the step that follows the **web_reg_async_attributes** step. For details on the **web_reg_async_attributes** step, see the [Function Reference \(Help > Function Reference\)](#).

Inserting a Comment

When VuGen inserts a **web_reg_async_attributes** step into a script, VuGen inserts an associated comment before the **web_reg_async_attributes** step. The inserted comment contains information about the associated asynchronous conversation, such as the conversation ID, the communication pattern (*push*, *poll*, or *long-poll*), a list of URLs that are part of the asynchronous communication, and list of callbacks implemented in the AsyncCallbacks.c extra file.

Notice that the step comment contains a TODO token. The TODO token indicates that you should check the relevant callback implementations in the AsyncCallbacks.c extra file.

For details on how an asynchronous conversation is terminated, see ["Defining the End of an Asynchronous Conversation" on the next page](#).

Example - web_reg_async_attributes

The sample code below shows a **web_reg_async_attributes** step that was added by VuGen. Notice that the **web_reg_async_attributes** step was added before a **web_url** step, and that the URL parameter in the **web_reg_async_attributes** step is the same as the URL parameter in the **web_url** step.

```
/* Added by Async CodeGen.  
ID=Poll_0  
ScanType = Recording  
  
The following urls are considered part of this conversation:  
http://your URL.com/content.php?messages  
  
TODO - The following callbacks have been added to AsyncCallbacks.c.  
Add your code to the callback implementations as necessary.  
Poll_0_RequestCB  
Poll_0_ResponseCB  
*/  
web_reg_async_attributes("ID=Poll_0",  
    "URL=http://your URL.com/content.php?messages",  
    "Pattern=Poll",  
    "RequestCB=Poll_0_RequestCB",  
    "ResponseCB=Poll_0_ResponseCB",  
    LAST);  
  
web_url("content.php",  
    "URL=http://your URL.com/content.php?messages",  
    "Resource=0",  
    "RecContentType=text/xml",  
    "Referer=http://your URL.com/",  
    "Snapshot=t2.inf",  
    "Node=HTML",  
    LAST);
```

Defining the End of an Asynchronous Conversation

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After VuGen scans a Vuser script for asynchronous communication, the Async tab of the Design Studio lists the asynchronous conversations that VuGen found in the script. A **web_stop_async** step is inserted into the Vuser script at the end of each asynchronous conversation that was detected. Use VuGen's Step Navigator to find the associated **web_stop_async** steps in the Vuser script.

Note: In some cases VuGen will not add a **web_stop_async** step at the end of an asynchronous conversation. This may occur when VuGen is not able to determine where the asynchronous conversation ends. This can occur when the asynchronous conversation was added due to a specific Async rule or when the asynchronous conversation was not ended during the recording. For details on Async rules, see ["Async Rules Overview" on page 402](#).

After VuGen has inserted a **web_stop_async** step into a Vuser script, make sure the **web_stop_async** step was added in the correct location in the script, that is – where the asynchronous conversation should end when the Vuser script runs.

In order to make sure the asynchronous conversation ends correctly when the script runs, it may be necessary to modify the details of the **web_stop_async** step in the Vuser script. For details, see ["Fine-Tuning the End of an Asynchronous Conversation" on the next page](#).

Note: All Async conversations are automatically terminated at the end of each iteration even if the **Simulate a new user each iteration** runtime option is disabled.

For details on how an asynchronous conversation is started, see ["Defining the Start of an Asynchronous Conversation" on page 388](#).

Example: web_stop_async

In the code sample below, VuGen added a **web_stop_async** step at the end of a *poll* conversation. In this example, the original polling steps are commented out, and the **lr_think_time** steps that separated them have been merged into one **lr_think_time** step in order to emulate the duration of the entire *poll* conversation.

```
/* Removed by Async CodeGen.
ID = Poll_0
*/
/*
web_url("content.php_7",
    "URL=http://your URL.com/content.php?messages",
    "Resource=0",
    "RecContentType=text/xml",
    "Referer=http://your URL.com/",
    "Snapshot=t8.inf",
    "Mode=HTML",
    LAST);
*/

lr_think_time(24);

web_stop_async("ID=Poll_0",
    LAST);
```

Using Asynchronous Request Thresholds

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

You can fine-tune some of VuGen's behavior when VuGen scans a Vuser script to locate asynchronous communication. You use VuGen's asynchronous request thresholds to fine-tune VuGen's behavior. Each of the thresholds is associated with only one of the types of asynchronous conversations: *push*, *poll*, or *long-poll*.

- **Asynchronous request thresholds for push conversations**

Minimum Response Size. Specify the minimum response content length (in bytes) for defining *push* asynchronous conversations. If the server sent less than the specified value, VuGen will not classify the conversation as a push-type asynchronous conversation.

Maximum Sub Message Size. Specify the maximum sub message size (in bytes) sent by the server for defining *push* asynchronous conversations. If the server sent a sub message of size greater than the specified value, VuGen will not classify the conversation as a push-type asynchronous conversation.

Minimum Number of Sub Messages. Specify the minimum number of sub messages for defining *push* asynchronous conversations. A push conversation in which less than the specified number of sub messages was sent by the server will not be classified by VuGen as a push-type asynchronous conversation.

- **Asynchronous request thresholds for poll conversations**

Interval Tolerance. Specify the interval tolerance (in milliseconds) for classifying *poll* asynchronous conversations. A conversation in which intervals differ from each other by more than the specified value will not be classified by VuGen as a poll-type asynchronous conversation.

- **Asynchronous request thresholds for long-poll conversations**

Maximum Interval. Specify the maximum interval (in milliseconds) between the end of one response and the start of a new request for classifying *long-poll* asynchronous conversations. A conversation in which a request starts more than the specified value after the end of the previous response will not be classified by VuGen as a long-poll type asynchronous conversation.

For details on the available asynchronous request thresholds, see ["Asynchronous Options Dialog Box" on page 406](#).

Fine-Tuning the End of an Asynchronous Conversation

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After VuGen scans a Vuser script for asynchronous communication, the Async tab of the Design Studio lists the asynchronous conversations that were found in the script. A **web_stop_async** step is inserted into the Vuser script at the end of each asynchronous conversation that was detected. In order to make sure that each asynchronous conversation ends correctly when the script runs, it may be necessary to perform one or more of the following tasks:

- Remove the **web_stop_async** step so that the asynchronous conversation will be terminated at the end of the iteration.
- Move the **web_stop_async** step to a location that is after an existing action step or an existing **lr_think_time** step, so the asynchronous conversation will end after that step is performed.
- Add an **lr_think_time** step before the **web_stop_async** step, or change the time parameter in an existing **lr_think_time** step. Make sure that think-time is enabled in the runtime settings. For details, see the **General > Think Time** view.
- Add a **web_sync** step to stop the asynchronous conversation after a specified parameter receives a value. Use the asynchronous conversation callbacks to make sure the parameter receives a value only when you want to end the conversation.

Correlating Asynchronous Vuser Scripts

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After modifying a Vuser script to enable it to emulate asynchronous communication, it may be necessary to perform correlation activities on the modified script. Due to asynchronous nature, dynamic values from asynchronous communication cannot be handled by Design Studio, and must be correlated manually.

You can search for dynamic values inside Response callbacks functions using the **lr_save_param_regexp** function. This function can be called from a callback to extract the necessary value from server response (**ResponseCB**) or response buffer (**ResponseBodyBufferCB**), and assign this value to a parameter. This parameter can then be used for correlations.

For details about the **lr_save_param_regexp** function, see the Function Reference (**Help > Function Reference**).

Implementing Callbacks

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After VuGen scans a Vuser script for asynchronous communication, the **Async** tab of the **Design Studio** lists the asynchronous conversations that were found in the script. For each asynchronous conversation found during the scan, VuGen adds the callback function signatures matching those declared in the **web_reg_async_attributes** step. The signatures are added to the AsyncCallbacks.c extra file.

The names of the callback functions start with the conversation ID of the asynchronous conversation. For example, the RequestCB callback for an asynchronous conversation with ID "LongPoll_0" will be **LongPoll_0_RequestCB**.

The names of the callback functions are declared in the **web_reg_async_attributes** step in the script.

The available callbacks are:

- **RequestCB**

This callback is called before a request is sent.

- **ResponseBodyBufferCB**

This callback is called when there is content in the response body buffer and at the end of the response body. This callback is generated by VuGen automatically for *push*-type conversations, but is available for *poll* and *long-poll* conversations as well.

- **ResponseCB**

This callback is called after every response is received in the conversation.

Example 1:

In the following sample code, the three callback functions are declared in the **web_reg_async_attributes** step.

```
/* Added by Async CodeGen.
ID=LongPoll_0
ScanType = Recording

The following urls are considered part of this conversation:
http://your URL.com/request.ashx?key=111111-11
http://your URL.com/request.ashx?key=111111-11
http://your URL.com/request.ashx?key=111111-11
http://your URL.com/request.ashx?key=111111-11

TODO - The following callbacks have been added to AsyncCallbacks.c.
Add your code to the callback implementations as necessary.
LongPoll_0_RequestCB
LongPoll_0_ResponseCB
*/
web_reg_async_attributes("ID=LongPoll_0",
    "URL= http://your URL.com/request.ashx?key=111111-11",
    "Pattern=LongPoll",
    "RequestCB=LongPoll_0_RequestCB",
    "ResponseCB=LongPoll_0_ResponseCB",
    LAST);
```

Example 2:

In the following sample code, the two callbacks are implemented in the AsyncCallbacks.c extra file.

```
int LongPoll_0_RequestCB()
{
    //enter your implementation for RequestCB() here

    //call web_util_set_request_url() here to modify polling url
    //web_util_set_request_url("<request_url>");

    //call web_util_set_request_body() here to modify request body:
    //web_util_set_request_body("<request body>");

    return WEB_ASYNC_CB_RC_OK;
}

int LongPoll_0_ResponseCB(
    const char *    aResponseHeadersStr,
    int             aResponseHeadersLen,
    const char *    aResponseBodyStr,
    int             aResponseBodyLen,
    int             aHttpStatusCode)
{
    //enter your implementation for ResponseCB() here

    return WEB_ASYNC_CB_RC_OK;
}
```

You can modify the callbacks to implement the required behavior. For details, see ["Modifying Callbacks" on the next page](#).

Example 3:

The following sample code shows an implementation of the ResponseHeader callback function, including the three arguments: HTTP Status code, Accumulated headers string and Accumulated headers string length.

```
int Push_0_ResponseHeadersCB(
    int aHttpStatusCode,
    const char * aAccumulatedHeadersStr,
    int aAccumulatedHeadersLen)
{
    //Enter your implementation for ResponseHeadersCB() here.

    Ir_output_message("Response status code is :[%d]",
aHttpStatusCode);
}
```

```
    lr_output_message("Response headers are :/n[%s]",  
aAccumulatedHeadersStr);  
  
    return WEB_ASYNC_CB_RC_OK;  
  
}
```

A sample of the output from the above callback function is shown below:

```
Response status code is :[200]  
Response headers are :  
[HTTP/1.1 200 OK  
Connection: close  
Date: Tue, 25 Jun 2013 09:03:33 GMT  
Server: Microsoft-IIS/6.0  
Content-Type: text/html  
Cache-control: private]
```

Modifying Callbacks

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

After VuGen scans a Vuser script for asynchronous communication, the **Async** tab of the **Design Studio** lists the asynchronous conversations that were found in the script. For each asynchronous conversation found during the scan, VuGen adds the required callback function declarations in the **AsyncCallbacks.c** file. To implement the required behavior, you can modify the callbacks that were added by VuGen. Modifying a callback includes:

Modifying the request URL in the RequestCB callback

In *poll* and *long-poll* conversations, requested URLs often change in each polling iteration. The change is usually determined by client-side logic, usually implemented by JavaScript that is executed by the browser. Parts of the URL may be determined by correlation of a known parameter, such as a session ID. For details, see ["Parsing URLs" on page 399](#).

Request URLs in an asynchronous conversation will be modified before the request is sent by using the RequestCB and the **web_util_set_request_url** function.

The following urls are considered part of this conversation:

```
http://www.example.com/example.aspx?message=helloaaa&iteration=1&timestamp=1324389551431  
http://www.example.com/example.aspx?message=hellobbb&iteration=2&timestamp=1324389555643  
http://www.example.com/example.aspx?message=helloccc&iteration=3&timestamp=1324389558664  
http://www.example.com/example.aspx?message=hellooddd&iteration=4&timestamp=1324389560113
```

Modifying the request body in the RequestCB callback

The request body in requests that are part of an asynchronous conversation may need to be modified before the request is sent. You use the RequestCB and the **web_util_set_request_body** util function to modify the request body.

Modifying the request body is useful in *poll* and *long-poll* conversations in which each new request requires a different request body.

```
//an example of a parametrized request body sent in the RequestCB.  
//the value of {request_body} may be set in the callback function,  
//or elsewhere in the script.  
web_util_set_request_body("{request_body}");
```

Each RequestCB that is generated by VuGen contains a commented snippet. You can "uncomment" the snippet in order to use the **web_util_set_request_body** util function.

If VuGen finds that different requests have different body values in the recorded conversation, the generated RequestCB will contain a comment that prompts you to check the recording in order to implement the request body sent in each request when the script runs.

```
//call web_util_set_request_body() here to modify request body:  
//web_util_set_request_body("<request body>");  
//TODO - use snapshot view to see examples of request bodies sent
```

Modifying the response in the ResponseCB callback

You can modify the response callback in an asynchronous conversation to verify validity of the responses, or to wait for a specific event. For example, you could check the response headers for each

response to determine if a specific value was received.

```
int Poll_0_ResponseCB(  
    const char *    aResponseHeadersStr,  
    int             aResponseHeadersLen,  
    const char *    aResponseBodyStr,  
    int             aResponseBodyLen,  
    int             aHttpStatusCode)
```

When the expected value has been received, you can use a **web_stop_async** step in the Action file to end the asynchronous conversation.

The following code sample provides an example for ending a synchronized conversation. In this example, in the **AsyncCallback.c** file, the scripts counts 10 iterations of the polling conversation, after which it creates a new parameter, **stopAsync**.



Example:

```
int Poll_0_ResponseCB(  
    ...{  
        //increment iteration counter for every response received.  
        static int iter = 0;  
        iter++;  
  
        //Once the desired number of responses has been reached,  
        //create and save the parameter.  
        if (iter > 10) {  
            lr_save_int(iter, "stopAsync");  
        }  
        return WEB_ASYNC_CB_RC_OK;  
    }  
}
```

In the Action file, the **web_sync** step uses the generated parameter, **stopAsync**, to end the conversation:



Example:

```
web_reg_async_attributes("ID=Poll_0","Pattern=Poll",  
    "URL=http://pumpkin:2080/nioamfpoll;AMFSessionId=6F8D6108-E309-38B2-3D65-  
963B431D0A38",  
    "PollIntervalMs=3000",  
    "RequestCB=Poll_0_RequestCB",  
    "ResponseCB=Poll_0_ResponseCB",  
    LAST);
```



```
web_custom_request("nioamfpoll;AMFSessionId=6F8D6108-E309-38B2-3D65-963B431D0A38_2",
    "URL=http://pumpkin:2080/nioamfpoll;AMFSessionId=6F8D6108-E309-38B2-3D65-963B431D0A38",
    "Method=POST",
    "Resource=0",
    "RecContentType=application/x-amf",
    "Referer=http://pumpkin:8081/lcds-samples/traderdesktop/traderdesktop.swf/[[DYNAMIC]]/3",
    "Snapshot=t11.inf",
    "Mode=HTML",
    "EncType=application/x-amf",

    "BodyBinary=\\x00\\x03\\x00\\x00\\x00\\x01\\x00\\x04null\\x00\\x02/6\\x00\\x00\\x00Z\\n\\x00\\x00\\x00\\x01\\x11\\n\\x07\\x07DSC\\x8D\\x02\\n\\x0B\\x01\\x01\\x06\\x01\\n\\x05\\tDSId\\x06I6F8D611E-DC1C-9D0C-2BBA-36CC2AB8633B\\x01\\x0C!\\xC0\\xBE\\xA6Z74\\xBE\\xC3\\xCF\\xC7\\xFA\\xE6\\xC3\\t\\xE2\\x92\\x01\\x06\\x01\\x01\\x04\\x02",
    LAST);

lr_think_time(30);

//suspend the script until the desired number
//of iterations have been performed.
web_sync("ParamCreated=stopAsync", "RetryIntervalMs=1000",
    "RetryTimeoutMs=300000", LAST);

web_stop_async("ID=Poll_0", LAST);
```

For more details about ending an asynchronous conversation, see ["Defining the End of an Asynchronous Conversation" on page 389](#).

Modifying callbacks in Flex Vuser scripts

For Flex asynchronous *polling* and *long-polling* conversations, the generated RequestCB in the **AsyncCallback.c** file contains a call to `web_util_set_formatted_request_body`, which sets an XML formatted request body for each request.



Example: `web_util_set_formatted_request_body("<AMFPacket AMF_version=\\\"3\\\">"`
`"<AMFHeaders />"`
`"<Messages>"`
`"<Message method=\\\"null\\\" target=\\\"/{Target_Poll_0}\\\">"`
`"..."`
`"</Message>"`
`"</Messages>"`
`"</AMFPacket>");`

After uncommenting the commented code in the TODO section, and adding your callback code, open the Script Design Studio to scan for correlations.

Note that code generation automatically parameterizes the Target parameter in the request body. It also generates code for automatically incrementing this parameter before each polling iteration.

The generated RequestCB also contains a reminder to ensure that the counter initialization parameter for **Target_Poll_0** in the Action file matches the target attribute in the first *Message* element in the first polling request.



Example: `lr_param_increment("Target_Poll_0", "{Target_Poll_0}");`

```
web_util_set_formatted_request_body("<AMFPacket AMF_version=\"3\">"
    "<AMFHeaders />"
    "<Messages>"
        "<Message method=\"null\" target=\"/{Target_Poll_0}\">"
        ...
```

In the Action file, make sure that you initialize the same polling parameter used in **AsyncCallbacks.c**. In the following segment from the Action file, the polling parameter, *Target_Poll_0*, matches the one used in **AsyncCallbacks.c**.



Example:

```
/* Initialize target parameter before sending first request. */
/* Notice that parameter will be incremented once before first request. */
lr_save_int(5, "Target_Poll_0");
```

For more details on using callback functions, see ["Implementing Callbacks" on page 392](#).

Parsing URLs



For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

URLs that are included in asynchronous conversations often include query strings that are derived in a variety of ways. These strings may include:

- Time-stamps
- Counters
- Complex strings

To enable a Vuser script to successfully perform asynchronous communication, VuGen must be able to recreate the required URLs.

- When the URL includes a time-stamp, VuGen is usually able to successfully create the required URL.
- When the URL includes a counter, VuGen is usually able to recreate the counter, but it may be necessary to manually initialize the counter in the script.
- When the URL includes more complex strings, the algorithms for generating the URLs must be manually added to the code in the Vuser script.

Example:

The sample code below shows a set of URLs that are part of a *long-poll* conversation. The URLs are included in the comment generated for a **web_reg_async_attributes** step:

```
The following urls are considered part of this conversation:  
http://www.example.com/example.aspx?message=helloaaa&iteration=1&timestamp=1324389551431  
http://www.example.com/example.aspx?message=hellobbb&iteration=2&timestamp=1324389555643  
http://www.example.com/example.aspx?message=helloccc&iteration=3&timestamp=1324389558664  
http://www.example.com/example.aspx?message=hellooddd&iteration=4&timestamp=1324389560113
```

If none of the parameters shown in the code sample above was found in VuGen's scan of the recorded Vuser script, the RequestCB implementation will contain a snippet that may be uncommented in order to set the URL for each response according to user defined code. For details, see ["Modifying Callbacks" on page 395](#).

```
//call web_util_set_request_url() here to modify request url:  
//web_util_set_request_url("<request url>");
```

If any or all of the parameters shown in the sample code above are found during VuGen's scan of the recorded Vuser script, the RequestCB implementation will contain the following:

- A comment prompting the user to call **web_util_set_request_url**. The comment will contain a parameterized version of the URL.
- For each *time-stamp* parameter found in the URL, a snippet for saving the time-stamp to a parameter.
- For each *counter* parameter found in the URL, a snippet for incrementing a counter parameter. A matching step for initializing the counter parameter will also be added to the Action file. The snippet will also contain examples of the URL token containing the counter parameter, as seen during the recording.
- For each *complex string* parameter found in the URL, a snippet for saving the string to a parameter. It is up to the user to generate the correct string to be saved to the parameter to be used in the URL. The snippet will also contain examples of the URL token that is considered an unknown parameter, as seen during the recording.

- A snippet for passing the parameterized version of the URL to the **web_util_set_request_url** function.

Example: A snippet containing the parameterized version of a URL.

```
//call web_util_set_request_url() here to modify polling url
//url is expected to be of the form:
//http://www.example.com/example.aspx?message={Unknown_LongPoll_0_0}
//&iteration={Counter_LongPoll_0_1}&timestamp={TimeStamp_LongPoll_0_2}
```

Example: A snippet prompting the user to set the value of an unknown parameter.

```
//TODO - implement parameter of type unknown: Unknown_LongPoll_0_0.
//Known examples for Unknown_LongPoll_0_0:
//message=[{"channel":"\\meta\\connect","connectionType":"long-polling","id":3,"clientId":"113fc44"}],
//message=[{"channel":"\\meta\\connect","connectionType":"long-polling","id":5,"clientId":"113fc44"}],
//message=[{"channel":"\\meta\\connect","connectionType":"long-polling","id":6,"clientId":"113fc44"}],
//message=[{"channel":"\\meta\\connect","connectionType":"long-polling","id":7,"clientId":"113fc44"}].
lr_save_string("[{"channel":"\\meta\\connect","connectionType":"long-polling","id":3,"clientId":"113fc44"}]",
"Unknown_LongPoll_0_0");
```

Example: A snippet for incrementing a counter parameter.

```
//TODO - check counter initialization for Counter_LongPoll_0_1 in Action file.
//Known examples for the token containing Counter_LongPoll_0_1:
//iteration=1, iteration=2, iteration=3, iteration=4 gPoll_0_1:
lr_param_increment("Counter_LongPoll_0_1", "{Counter_LongPoll_0_1}");
```

Example: A snippet for initializing a counter parameter.

```
lr_save_int(0, "Counter_LongPoll_0_1");
```

Example: A snippet for saving a timestamp parameter.

```
web_save_timestamp_param("TimeStamp_LongPoll_0_2", LAST);
```

Example: A snippet for passing the parameterized version of a URL to the **web_util_set_request_url** function.

```
//once all parameters have been assigned, copy them to the updated url,  
//and call web_util_set_request_url() with the updated url:  
web_util_set_request_url("http://www.example.com/example.aspx?message={Unknown_LongPoll_0_0}  
&iteration={Counter_LongPoll_0_1}&timestamp={TimeStamp_LongPoll_0_2}");
```

Async Rules Overview

Note: For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording"](#) on page 422.

In some cases, when VuGen performs an Async scan, VuGen may fail to correctly identify some of the asynchronous conversations that are included in the Vuser script. In other cases, VuGen may erroneously classify regular synchronous steps as part of asynchronous conversations. To help rectify both of these scenarios, you can define Async rules to determine how requests to specified URLs are classified during an Async scan.

Async rules can be positive or negative.

- **Positive:** When VuGen fails to identify asynchronous conversations that are included in a Vuser script, implement a positive Async rule to enable VuGen to identify the asynchronous behavior.

Scenario: VuGen does not identify URLs under **http://www.true-async.com/push_example.aspx** as push asynchronous conversations, and you know that they are part of push asynchronous conversations. Add a positive rule to enable VuGen to correctly identify the push asynchronous conversations. When you regenerate the script, the Async scan will apply the added rule, and all URLs that start with **http://www.true-async.com/push_example.aspx** will be included as part of push asynchronous conversations.

- **Negative:** When VuGen erroneously classifies regular synchronous steps as part of an asynchronous conversation, implement a negative Async rule to prevent VuGen from erroneously identifying asynchronous behavior.

Scenario: VuGen identifies all URLs under **http://www.not-async.com/** as asynchronous poll conversations. You know that these are not asynchronous conversations. Implement a negative Async rule to prevent VuGen from erroneously identifying asynchronous behavior. When you regenerate the script, the Async scan will apply the added rule so that all URLs that start with **http://www.not-async.com/** will not be classified as part of asynchronous conversations.

For details on how to implement Async rules, see ["Adding Async Rules"](#) below.


Adding Async Rules

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording"](#) on page 422.

When VuGen scans a Vuser script after recording or regenerating the script, VuGen may fail to identify asynchronous conversations that are included in the Vuser script. In other cases, VuGen may erroneously classify a regular synchronous step as part of an asynchronous conversation.

You can define Async rules that determine how requests to specified URLs will be classified during an Async scan. Async rules may be positive or negative.

Adding a positive Async rule

1. Select **Record > Recording Options > General > Code Generation** and then click **Async Options**. The Asynchronous Options dialog box opens.
2. Under **Asynchronous Regular Expressions**, click **Add Async Rule** . The Add Rule dialog box opens.
3. From the **Type** list, select **Push**, **Poll**, or **Long Poll**, as required.
4. In **URL Regular Expression**, enter a regular expression for URLs that should be considered part of asynchronous conversations.

Special characters that you can include in a regular expression:

. Any single character
* Zero or more
+ One or more
? Zero or one
^ Beginning of line
\$ End of line
\n Line break
\r Carriage return
[] Any one character in the set
[^] Any one character not in the set
\w Word character
\W Non-word character
\d Decimal digit
\D Non-decimal digit
Or
\ Escape special character

To include characters such as “?” and “+” in the regular expression, insert a backslash “\” before the required character.


5. Click **OK**. The new rule appears in the list of Async rules for the Vuser script.

When you regenerate the script:

- For each push conversation that includes a URL that matches the regular expression, VuGen inserts asynchronous API functions into the Vuser script, but does not remove any of the recorded code from the Vuser script.
- For each polling or long-polling conversation that includes a URL that matches the regular expression, VuGen inserts asynchronous API functions into the Vuser script, and may remove steps or step parameters from the generated Vuser script. VuGen removes steps or step parameters in cases where the relevant URLs will be requested by running the inserted asynchronous functions - and not by running the original steps that have been removed.

For further details, see ["How VuGen Modifies a Vuser Script for Asynchronous Communication" on page 385](#).

Adding a negative Async rule

1. Select **Record > Recording Options > General > Code Generation** and then click **Async Options**. The Asynchronous Options dialog box opens.
2. Under **Synchronous Regular Expressions**, click **Add Async Rule** . The **Add Asynchronous Rule** dialog box opens.
3. From the **Rule Type** list, select **Not Async**.
4. In **URL Regular Expression**, enter a regular expression for URLs that should not be considered part of asynchronous conversations.
5. Click **OK**. The new rule appears in the list of Async rules for the Vuser script.


When you regenerate the script, steps that contain URLs that match the regular expression will not be included in asynchronous conversations.

See also:

- ["Async Rules Overview" on page 402](#)

Async Tab [Design Studio]

The Async tab of the Design Studio lists all the occurrences of asynchronous communication that VuGen detected in the Vuser script.

To access	<ul style="list-style-type: none">• Select Design > Design Studio, and then click the Async tab.• Click the  Design Studio button on the VuGen toolbar, and then click the Async tab.
------------------	--

Important information	<ul style="list-style-type: none"> The Design Studio button is enabled only when you display a recorded Vuser script in the Solution Explorer. The Async tab enables you to only view the asynchronous communication that is included in the Vuser script - you cannot edit any of the asynchronous details from the Async tab. Changes to the asynchronous details must be made in the Vuser script.
Relevant tasks	"Create an Asynchronous Vuser Script" on page 381

User interface elements are described below:

UI Element	Description
Type	<p>Indicates the origin of the asynchronous code in the Vuser script:</p> <p>Record. The asynchronous code was added by VuGen during an Async scan that was performed after recording or regenerating the Vuser script.</p> <p>Rule. The asynchronous code was added by VuGen due to a specific rule in the Async rules file.</p> <p>Manual. The asynchronous code was manually added by a user.</p>
Action	The section of the Vuser script in which the asynchronous behavior is located.
Occurrences	<ul style="list-style-type: none"> For push-type conversations, Occurrences is always 1. For poll and long-poll conversations, Occurrences indicates the number of steps or extra resource attributes that were removed [commented-out] by VuGen during the Async scan of the Vuser script.
Status	Always has the value Applied .
Async Type	The type of the asynchronous behavior that was detected: Push , Poll , or Long-Poll .
URL	The URL in the web_reg_async_attributes step that starts the asynchronous conversation.
Filter	Select which asynchronous conversations to display in the conversation list.
Details	Expands the dialog box to show details about the selected asynchronous conversation.
Name	Always has the value web_reg_async_attributes .
Line	The line in the Vuser script that contains the web_reg_async_attributes step.
Action Name	The section of the Vuser script in which the asynchronous behavior is located.

UI Element	Description
Description	The comment in the Vuser script that precedes the web_reg_async_attributes step.
Occurrences in Snapshot	<ul style="list-style-type: none"> For push-type conversations, displays the response body. For poll and long-poll conversations, displays HTTP attributes associated with the asynchronous conversation.
Options	Opens the Asynchronous Request Thresholds dialog box.

Asynchronous Options Dialog Box

This dialog box enables you to fine-tune some of VuGen's behavior when VuGen scans a Vuser script to locate asynchronous communication.

To access	Record > Recording Options > General > Code Generation and then click Async Options .
------------------	--

User interface elements are described below:

UI Element	Description
Minimum Response Size	Specify the minimum size (in kilobytes) of a server response for defining <i>push</i> asynchronous conversations. If the server sent less than the specified value, VuGen will not classify the conversation as a push-type asynchronous conversation.
Maximum Sub Message Size	Specify the maximum sub message size (in kilobytes) sent by the server for defining <i>push</i> asynchronous conversations. If the server sent a sub message of size greater than the specified value, VuGen will not classify the conversation as a push-type asynchronous conversation.
Minimum Number of Sub Messages	Specify the minimum number of valid sub messages for defining <i>push</i> asynchronous conversations. A push conversation in which less than the specified number of valid sub messages was sent by the server will not be classified by VuGen as a push-type asynchronous conversation.
Interval Tolerance	Specify the interval tolerance (in milliseconds) for classifying <i>poll</i> asynchronous conversations. A conversation in which intervals differ from each other by more than the specified value will not be classified by VuGen as a poll-type asynchronous conversation.
Maximum Interval	Specify the maximum interval (in milliseconds) between the end of one response and the start of a new request for classifying <i>long poll</i> asynchronous conversations. A conversation in which a request starts more than the specified value after the end of the previous response will not be classified by VuGen as a long-poll type asynchronous conversation.

Asynchronous Regular Expressions Table	
<Activate Rule>	A check box indicating whether or not the rule is activated. To select all rules, select the check box in the toolbar.
Rule Type	Push , Poll , and Long Poll are positive asynchronous rules. Not Async is a negative rule. For details on the asynchronous rule types, see "Async Rules Overview" on page 402 .
Regular Expression	A regular expression for URLs that should be considered part of asynchronous conversations. For a list of the special characters that you can include in a regular expression, see "Adding Async Rules" on page 402 .

Asynchronous Example - Poll

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

The following example describes a Vuser script that includes a poll asynchronous conversation. The application that is emulated by the Vuser is a demo of a messaging service that sends updates on demand. The browser displays the page, and sends requests for updates at intervals of approximately 5 seconds.

```
web_url("message.txt",
    "URL=http://example.com/content/message.txt?key=0",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t4.inf",
    "Mode=HTML",
    LAST);

lr_think_time(4);

web_url("message.txt_2",
    "URL=http://example.com/content/message.txt?key=1",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t5.inf", |
    "Mode=HTML",
    LAST);

lr_think_time(5);

web_url("message.txt_3",
    "URL=http://example.com/content/message.txt?key=2",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t6.inf",
    "Mode=HTML",
    LAST);

lr_think_time(4);

web_url("message.txt_4",
    "URL=http://example.com/content/message.txt?key=3",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t7.inf",
    "Mode=HTML",
    LAST);

lr_think_time(5);

web_url("message.txt_5",
    "URL=http://example.com/content/message.txt?key=4",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t8.inf",
    "Mode=HTML",
    LAST);
```

Note: You can modify VuGen's asynchronous request thresholds to assist VuGen in finding poll-type conversations. For details, see ["Using Asynchronous Request Thresholds" on page 391](#).

The above script was generated by VuGen after the required business processes were recorded. An asynchronous scan was not performed on the script after the script was generated. Notice that the script contains a series of **web_url** functions with a repeating URL, namely:

http://example.com/content/message.txt. These **web_url** functions are separated by **lr_think_time** functions, indicating that the **web_url** functions repeat at intervals of approximately 5 seconds.

When the Vuser script runs, requests for **http://example.com/content/message.txt** should be sent repeatedly until the script is finished. Additionally, these requests should be sent in parallel (simultaneously) with other actions performed in the Vuser script.

After VuGen performed a scan for asynchronous communications on the script, the script looks as follows:

```

- /* Added by Async CodeGen.
  ID=Poll_0
  ScanType = Recording

  The following URLs are considered part of this conversation:
  http://example.com/content/message.txt?key=0
  http://example.com/content/message.txt?key=1
  http://example.com/content/message.txt?key=2
  http://example.com/content/message.txt?key=3
  http://example.com/content/message.txt?key=4

  TODO - The following callbacks have been added to AsyncCallbacks.c.
  Add your code to the callback implementations as necessary.
  Poll_0_RequestCB
  Poll_0_ResponseCB
- */
  web_reg_async_attributes("ID=Poll_0",
    "Pattern=Poll",
    "URL=http://example.com/content/message.txt?key=0",
    "PollIntervalMs=4900",
    "RequestCB=Poll_0_RequestCB",
    "ResponseCB=Poll_0_ResponseCB",
    LAST);

  lr_save_int(0, "Counter_Poll_0_0");

  web_url("message.txt",
    "URL=http://example.com/content/message.txt?key=0",
    "Resource=0",
    "RecContentType=text/plain",
    "Snapshot=t4.inf",
    "Mode=HTML",
    LAST);

+ /* Removed by Async CodeGen.
+   /*
+ /* Removed by Async CodeGen.
+   /*
+ /* Removed by Async CodeGen.
+   /*
+ /* Removed by Async CodeGen.
+   /*
+ /* Removed by Async CodeGen.
+   /*

  lr_think_time(20);

  web_stop_async("ID=Poll_0",
    LAST);

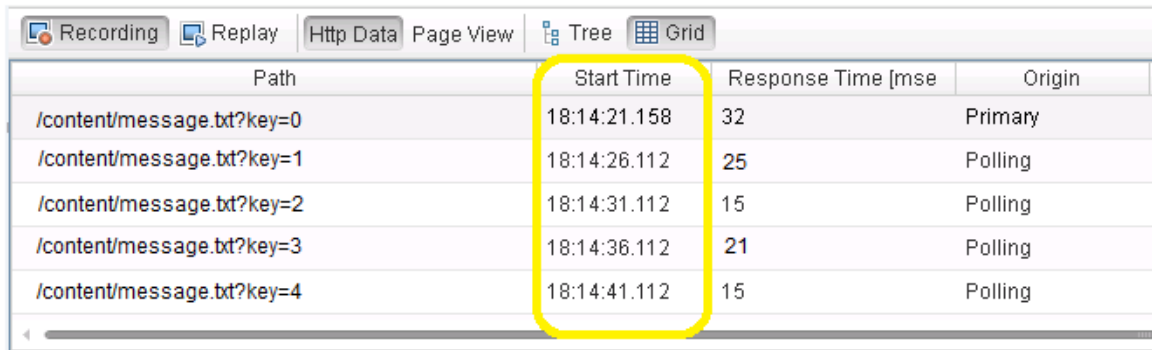
```

Notice that a **web_reg_async_attributes** function has been added to the script before the first **web_url** function that calls **<http://example.com/content/message.txt>**, and that a **web_stop_async** function has been added after the last **web_url** function that calls the same URL.

Except for the first call to **http://example.com/content/message.txt**, all other **web_url** functions that call the same URL have been commented-out by VuGen.

Notice that the **lr_think_time** function has been replaced by the **PollIntervalMs** argument for **web_reg_async_attributes**.

The Snapshot pane for the remaining **web_url** function shows that the snapshots for the removed **web_url** functions now have Origin = Polling, and that they start at intervals of 5 seconds.



Path	Start Time	Response Time [msec]	Origin
/content/message.txt?key=0	18:14:21.158	32	Primary
/content/message.txt?key=1	18:14:26.112	25	Polling
/content/message.txt?key=2	18:14:31.112	15	Polling
/content/message.txt?key=3	18:14:36.112	21	Polling
/content/message.txt?key=4	18:14:41.112	15	Polling

Asynchronous Example - Push

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

The following example describes a Vuser script that is developed to emulate a browser displaying an application that utilizes push-type asynchronous conversations. The application is a demo of a “stock quote” page. The browser shows the page with the stock values, and then sends a request and receives a response with updated stock values. The request remains open until it is closed by the user. For as long as the page is displayed, the server will continue to send sub-messages as part of the response - whenever the server has an update for the displayed stocks. Whenever such a sub-message is received by the client, the client displays the updated stock values.

Note: You can modify VuGen's asynchronous request thresholds to assist VuGen in finding push-type conversations. For details, see ["Using Asynchronous Request Thresholds" on page 391](#).

Name	Price	Time	Change	Bid Size	Bid	Ask	Ask Size	Min	Max	Ref.
Anduct	3.09	10:32:54	1.64	5000	3.09	3.1	35500	2.48	3.64	3.04
Ations Europe	17.92	10:33:05	11.37	86000	17.92	17.98	1000	12.8	19.33	16.09
Bagies Consulting	6.43	10:33:04	-10.57	4500	6.43	6.44	8500	5.74	8.64	7.19
BAY Corporation	3.2	10:32:03	-11.84	37500	3.2	3.21	63500	3.13	4.28	3.63
CON Consulting	6.8	10:33:05	-10.64	24000	6.8	6.83	51000	6.23	9.08	7.61
Corcor PLC	2.58	10:32:54	12.17	61000	2.58	2.59	42000	1.87	2.7	2.3
CVS Asia	13.48	10:33:06	-12.41	94500	13.45	13.48	11000	12.38	18.34	15.39
Datio PLC	5.81	10:33:00	9.41	42500	5.8	5.81	1500	4.4	6.34	5.31
Dentems	4.87	10:33:05	0.2	12000	4.86	4.87	35500	3.91	5.67	4.86
ELE Manufacturing	7.19	10:33:01	-5.51	13500	7.19	7.2	51000	6.25	9.09	7.61

If you attempt to run a script that calls a push url - without first performing an asynchronous scan - the replay will halt while waiting for the response to the highlighted request. After two minutes, VuGen will display an error similar to the following, in the Replay log:

```
Action.c(140): Error -27782: Timeout (120 seconds) exceeded while waiting to receive
data for URL "http://push.example.com" [MsgId: MERR-27782]
```

The error indicates that the response never finished.

Regenerating the script with Async Scan enabled will create a script similar to the following:

```

/* Added by Async CodeGen.
ID=Push_0
ScanType = Recording

The following urls are considered part of this conversation:
http://push.example.com/STREAMING\_IN\_PROGRESS?LS\_session=5343716d5eb050c64T2253451&LS\_phase=4903&LS\_domain=lights,"

TODD - The following callbacks have been added to AsyncCallbacks.c.
Add your code to the callback implementations as necessary.
Push_0_RequestCB
Push_0_ResponseBodyBufferCB
Push_0_ResponseCB
*/
web_reg_async_attributes("ID=Push_0",
    "URL=http://push.example.com/STREAMING\_IN\_PROGRESS?LS\_session={CorrelationParameter}&LS\_phase=4903&LS\_domain=",
    "Pattern=Push",
    "RequestCB=Push_0_RequestCB",
    "ResponseBodyBufferCB=Push_0_ResponseBodyBufferCB",
    "ResponseCB=Push_0_ResponseCB",
    LAST);

web_url("STREAMING_IN_PROGRESS",
    "URL=http://push.example.com/STREAMING\_IN\_PROGRESS?LS\_session={CorrelationParameter}&LS\_phase=4903&LS\_domain=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=http://www.app.example.com/GWT\_StocklistDemo\_Basic/lightstreamer/lengine.html",
    "Snapshot=t13.inf",
    "Node=HTML",
    LAST);

web_custom_request("control.js",
    "URL=http://push.example.com/lightstreamer/control.js",
    "Method=POST",
    "Resource=0",
    "RecContentType=text/plain",
    "Referer=http://push.example.com/ajax\_frame.html?phase=594&domain=example.com&",
    "Snapshot=t14.inf",
    "Node=HTML",
    "Body=LS_session={CorrelationParameter}&LS_table=1&LS_win_phase=19&LS_req_phase=311&LS_op=add&LS_mode=MERGE&LS_id=item1",
    LAST);
/* Added by Async CodeGen.
ID = Push_0
*/
web_stop_async("ID=Push_0",
    LAST);

return 0;
}

```

Notice that a **web_reg_async_attributes** function has been added before the **web_url** function that starts the *push* conversation, and that a **web_stop_async** function has been added after the last action step in the script. The script will now run successfully. The *push* conversation will remain active – running in parallel with the other script functions – until the **web_stop_async** function, or until the end of the script is reached.

Note that during the Async scan, VuGen did not remove (comment-out) any of the generated code in the Vuser script.

Asynchronous Example - Long-Poll

For a list of protocols that support asynchronous communication, see ["Protocol Support for Async, IPv6, and 64-bit Recording" on page 422](#).

The following example describes a Vuser script that emulates an application that implements a *long-poll* asynchronous conversation. The application is a demo of a “chat” page. A browser shows the chat page, and sends a request that remains open until a new message is sent to the chat by another user. After such a message is sent:

- The response is finished.
- The new message is shown in the browser.
- The browser sends another request in order to listen for the next message sent to the chat.



Note: You can modify VuGen's asynchronous request thresholds to assist VuGen in finding long-poll type conversations. For details, see ["Using Asynchronous Request Thresholds" on page 391](#).

The following is the Vuser script that VuGen generated after recording the application - before an asynchronous scan was performed. The script contains a series of **web_url** functions with similar URLs. Since new requests are sent as soon as the previous response is finished, no **lr_think_time** functions are added between the **web_url** functions. This helps to indicate that this is a *long-poll* conversation and not a *poll* conversation.

When the Vuser script runs, requests to the chat application should be sent repeatedly every time a response from the chat application is finished. In addition, requests should be sent in parallel (simultaneously) with other actions performed in the script.

After VuGen performs an asynchronous scan on the script, the modified script looks as follows:

```

/* Added by Async CodeGen.
ID=LongPoll_0
ScanType = Recording

The following urls are considered part of this conversation:
http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D
http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D
http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D
http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D

TODO - The following callbacks have been added to AsyncCallbacks.c.
Add your code to the callback implementations as necessary.
LongPoll_0_RequestCB
LongPoll_0_ResponseCB
- */
web_reg_async_attributes("ID=LongPoll_0",
    "URL=http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D",
    "Pattern=LongPoll_1",
    "RequestCB=LongPoll_0_RequestCB",
    "ResponseCB=LongPoll_0_ResponseCB",
    LAST);

web_url("request.ashx_3",
    "URL=http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t4.inf",
    "Node=HTML",
    LAST);

/* Removed by Async CodeGen.
ID = LongPoll_0
- */
/* http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D
web_url("request.ashx_4",
    "URL=http://example.com/request.ashx?key=111111-1111-1111-1111-111111token=123488858&message=%5B%5D",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t5.inf",
    "Node=HTML",
    LAST);
- */

/* Removed by Async CodeGen.

```

Notice that a **web_reg_async_attributes** function has been added before the first **web_url** function that calls the chat application.


Except for the first call the chat application, all other **web_url** functions that call similar URLs have been commented out.

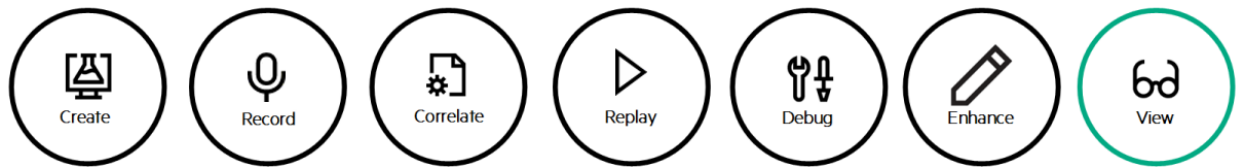
The Snapshot pane for the remaining **web_url** function shows that the snapshots for the removed steps now have **origin = Polling**.

The response times vary greatly as the responses arrive only when another user has sent a chat message. This helps to indicate that this is a long-poll conversation, and not a poll conversation.

Viewing Replay Results

Developing a Vuser script includes the steps shown below. This topic provides an overview of the seventh step, viewing the results of the replay of a Vuser script.

 Select an image to learn more.



To assist with debugging a Vuser script, you can view a Replay Summary report or a Business Process report after replaying the script. The Replay Summary report is useful typically during the scripting process, whereas the Business Process report provides higher level information or a business process view of the script.

- **Replay Summary report.** Summarizes the results of your script run. VuGen generates the report during the Vuser script execution and you view the report when script execution is complete. For details see ["Replay Summary Pane" on page 108](#).
- **Business Process report.** A customizable report that can include the replay summary, details of the transactions, rendezvous points, and parameters in the script, and additional information about the script replay. For details, see ["Create a Business Process Report" on page 140](#).

VuGen Protocols

VuGen enables you to record a variety of protocols, each suited to a particular load testing environment or topology and resulting in a specific type of Vuser script. For example, you can use a Web - HTTP/HTML Vuser Script to emulate users operating Web browsers. You can use FTP Vusers to emulate an FTP session. The various Vuser technologies can be used alone or together to create effective load tests.

The following table lists the available Vuser protocols, and a brief description of each protocol.

Note:

- Protocols with links have expanded information.
- When running tests using protocol-specific Vuser scripts, make sure you have a global license or the required license for your protocol.

Protocol	Description
.NET	Supports the recording of Microsoft .NET client-server technologies.
Ajax (Click & Script)	An acronym for Asynchronous JavaScript and XML. Ajax (Click & Script) uses asynchronous HTTP requests, allowing Web pages to request small bits of information instead of whole pages.
C Vuser	A generic virtual user which uses the standard C library.
Citrix ICA	A remote access tool, allowing users to run specific applications on external machines.
COM/DCOM	Component Object Model (COM) - a technology for developing reusable software components.
(DNS) Domain Name Resolution	The DNS protocol is a low-level protocol that allows you to emulate the actions of a user working against a DNS server. The DNS protocol emulates a user accessing a Domain Name Server to resolve a host name with its IP address. Only replay is supported for this protocol—you need to manually add the functions to your script.
Flex	Flex is an application development solution for creating Rich Internet Applications (RIAs) within the enterprise and across the Web. Action Message Format (AMF), is a Macromedia proprietary protocol that allows Flash Remoting binary data to be exchanged between a Flash application and an application server over HTTP.

Protocol	Description
FTP (File Transfer Protocol)	File Transfer Protocol - a system which transfers files from one location to another over a network. The FTP protocol is a low-level protocol that allows you to emulate the actions of a user working against an FTP server.
IMAP (Internet Messaging)	Internet Message Application - a protocol which enables clients to read email from a mail server.
Java over HTTP	Designed to record java-based applications and applets. It produces a Java language script using web functions. This protocol is distinguished from other Java protocols in that it can record and replay Java remote calls over HTTP.
Java Record Replay	Common Java recorder.
Java Vuser	Java programming language with protocol level support.
LDAP (Listing Directory Service)	An Internet protocol designed to allow email applications to look up contact information from a server.
MAPI (Microsoft Exchange)	Messaging Application Programming Interface designed to allow applications to send and receive email messages.
Web - HTTP/HTML (for mobile applications)	Enables the recording of mobile native applications.
MQTT	Enables lightweight, publish/subscribe messaging that is ideal for Internet of Things (IoT) and machine-to-machine (M2M) communications, as well as mobile applications where bandwidth and battery power are at a premium.
ODBC	Open Database Connectivity - a protocol providing a common interface for accessing databases.
Oracle - 2 Tier	Oracle database using a standard 2-tier client/server architecture.
Oracle - Web	The Oracle Applications interface that performs actions over the Web. This Vuser type detects actions on both the API and Javascript levels.
Oracle NCA	Oracle 3-tier architecture database consisting of Java client, Web server and database.
POP3 (Post Office Protocol)	A protocol designed to allow single computers to retrieve email from a mail server.

Protocol	Description
RDP (Remote Desktop Protocol)	A remote access tool using the Microsoft Remote Desktop Connection to run applications on an external machine.
RTE (Remote Terminal Emulator)	Emulation of users who submit input to, and receive output from, character-based applications.
SAP GUI	An Enterprise Resource Planning system to integrate key business and management processes using the SAP GUI client for Windows.
SAP - Web	An Enterprise Resource Planning system to integrate key business and management processes using the SAP Portal or Workplace clients.
Siebel - Web	A Customer Relationship Management Application.
SMP (SAP Mobile Platform)	A protocol for recording actions on a mobile SAP application.
SMTP (Simple Mail Protocol)	Simple Mail Transfer Protocol - a system for distributing mail to a particular machine.
"Teradici PCoIP (PC over IP) Protocol" on page 661	Supports testing on the Teradici platform.
TruClient - Mobile Web	Enables the recording of mobile browser based applications using the TruClient technology.
TruClient - Native Mobile	Records native mobile applications using the TruClient technology. For details, select the relevant version in the Mobile Center Help and see the Performance Testing section.
TruClient - Web	An advanced protocol for modern JavaScript-based applications emulating user activity within a Web browser. Scripts are developed interactively from within a Web browser. For details, go to the TruClient Help Center (select the relevant version).
Web - HTTP/HTML	Emulation of communication between a browser and Web server on an HTTP or HTML level.
Web Services	Web Services are a programmatic interface for applications to communicate with one another over the World Wide Web.

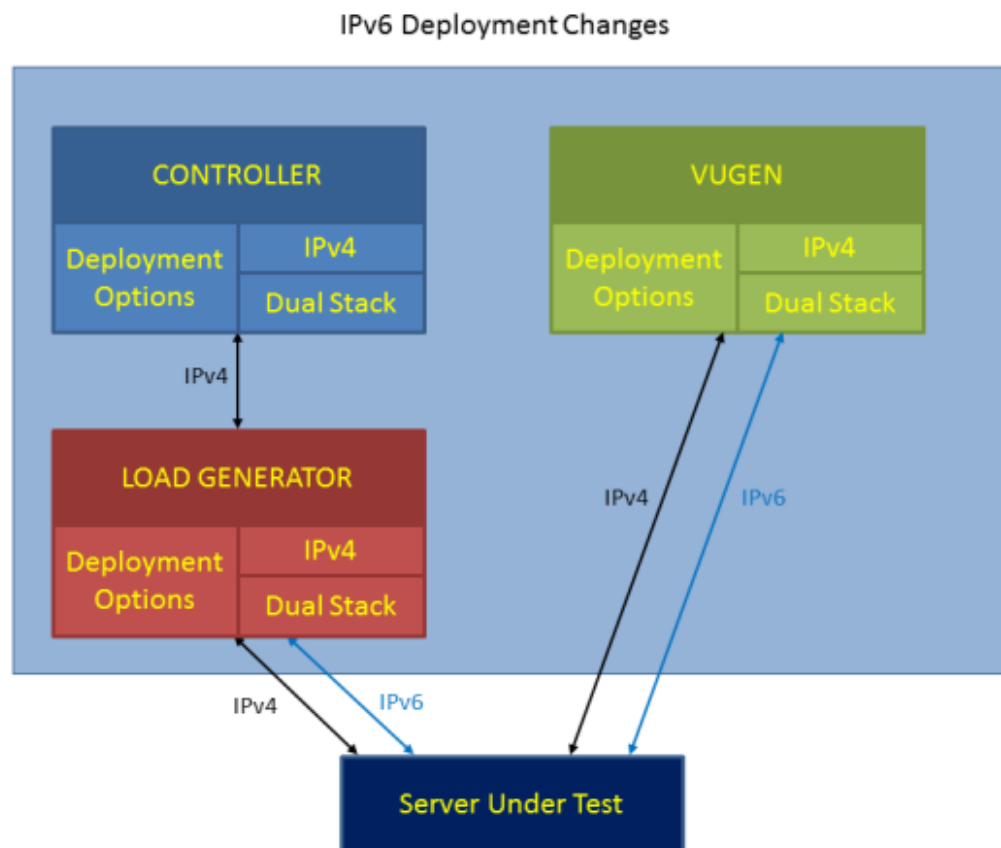
Protocol	Description
Windows Sockets	The standard network programming interface for the Windows platform.

IPv6 Support

Virtual User Generator enables you to test IPv6 based applications in addition to IPv4 based ones. Script recording supports recording for both IPv4 and IPv6 simultaneously. The code that is generated is non-IP specific. With the exception of Web HTTP/HTML protocols, users are unaware which IP version is being used when replaying the script in a load test. Web HTTP protocols have a Runtime setting that allows you to choose between IPv4 and IPv6 for the replay.

IPv6 Deployment

The internal Virtual User Generator communication between the Controller and Load Generators uses IPv4 communication. To record and replay in both IPv4 and IPv6, install both VuGen and Load Generator on IPv6-enabled computers, as shown in the diagram below.



For more details about IPv6 related changes, see the [Preferences View - Internet Protocol](#).

Protocols Supported

For a list of the supported protocols, see the [System Requirements](#). You can also find information here: ["Protocol Support for Async, IPv6, and 64-bit Recording" on the next page](#)

Protocol Support Limitations

Support for IPv6 is available with the following limitations:

Web HTTP protocol

- FTP from Web is not supported
- Web Breakdown is not supported
- Kerberos is not supported
- Spoofing from Web is not supported
- PAC file is not supported

Webtrace

- IPv6 Webtrace is not supported on 6to4 outgoing network interfaces.
- IPv6 webtrace does not support RawSocket mode

General limitations

- Replay failures may occur because of a IPv4/IPv6 switch between recording and replaying.

Troubleshooting

Problem:

IPv6_webtrace fails to run from a command line with error "webtrace_send_probe_failed". This is caused by an incorrectly installed load generator.

Solution:

1. Uninstall LoadRunner or the incorrectly installed load generator.
2. Check if the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SRPACKET key was removed by the uninstallation process (manually remove it if necessary).
3. Remove all files and folder in the **C:\ HPE\LoadGenerator** folder left by the uninstallation process.
4. Restart the machine.
5. Reinstall LoadRunner or the load generator.
6. Check if the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SRPACKET\ImagePath value points to the correct path (packet_amd64.sys driver).

Protocol Support for Async, IPv6, and 64-bit Recording

Note:

- The table below includes only the protocols that support at least one of the options.
- For all protocols that support asynchronous sessions, recording will be applied only to web_* steps.

Protocol	Async	IPv6	64-bit recording
Ajax Click & Script	N/A	✓	N/A
DNS	N/A	✓	N/A
Flex AMF	✓	✓	N/A
FTP	N/A	✓	✓
IMAP	N/A	✓	N/A
Java over HTTP	N/A	✓	✓
Java Record Replay	N/A	N/A	✓
Java Vuser	N/A	N/A	✓
LDAP	N/A	N/A	✓
MQTT	N/A	✓	N/A

Protocol	Async	IPv6	64-bit recording
.NET	N/A	N/A	✓
Oracle - 2 Tier	N/A	N/A	✓
Oracle NCA	N/A	✓	✓
Oracle - Web	N/A	✓	✓
POP3	N/A	✓	N/A
RTE	N/A	✓	N/A
RDP	N/A	✓	✓
SAP GUI	N/A	✓	N/A
Siebel - Web	N/A	N/A	✓
SMTP	N/A	✓	N/A
TruClient - Mobile Web	N/A	✓	N/A
TruClient - Web	N/A	✓	N/A
Web - HTTP/HTML (including HTTP/2)	✓	✓	✓

Protocol	Async	IPv6	64-bit recording
Web Services	✓	✓	✓
Windows Sockets (multi-protocol)	N/A	✓	✓

Ajax - Click & Script Protocol



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see ["Web - HTTP/HTML Protocol" on page 664](#) or go to the [TruClient Help Center](#) (select the relevant version).

Ajax (Asynchronous JavaScript and XML) represents a group of technologies for creating interactive Web applications. With Ajax, web pages exchange small packets of data with the server, instead of reloading an entire page. This reduces the amount of time that a user needs to wait when requesting data. It also increases the interactive capabilities and enhances the usability.

Using Ajax, developers can create fast Web pages using Javascript and asynchronous server requests. The requests can originate from user actions, timer events, or other predefined triggers.

Ajax components, also known as Ajax controls, are GUI based controls that use the Ajax technique—they send a request to the server when a trigger occurs.

For example, a popular Ajax control is a **Reorder List** control that lets you drag components to a desired position in a list. VuGen's support for Ajax implementation is based on Microsoft's ASP.NET Ajax Control Toolkit formerly known as Atlas.

Ajax (Click & Script) Protocol Overview



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.

- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see "[Web - HTTP/HTML Protocol](#)" on page 664 or go to the [TruClient Help Center](#) (select the relevant version).

Ajax (Asynchronous JavaScript and XML) represents a group of technologies for creating interactive Web applications. With Ajax, web pages exchange small packets of data with the server, instead of reloading an entire page. This reduces the amount of time that a user needs to wait when requesting data. It also increases the interactive capabilities and enhances the usability.

Using Ajax, developers can create fast Web pages using Javascript and asynchronous server requests. The requests can originate from user actions, timer events, or other predefined triggers.

Ajax components, also known as Ajax controls, are GUI based controls that use the Ajax technique—they send a request to the server when a trigger occurs.

For example, a popular Ajax control is a **Reorder List** control that lets you drag components to a desired position in a list. VuGen's support for Ajax implementation is based on Microsoft's ASP.NET Ajax Control Toolkit formerly known as Atlas.



See also:

- For an overview on the Click and Script protocols, see "[Click & Script Protocols - Overview](#)" on page 457.

Ajax (Click & Script) Supported Frameworks

Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see "[Web - HTTP/HTML Protocol](#)" on page 664 or go to the [TruClient Help Center](#) (select the relevant version).

The supported frameworks for Ajax Click & Script functions are:

- Atlas 1.0.10920.0/ASP.NET Ajax—All controls
- Scriptaculous 1.8—Autocomplete, Reorder List, and Slider

VuGen supports the following frameworks at the engine level. This implies that VuGen will create standard Click & Script steps, but not Ajax specific functions:

- Prototype 1.6
- Google Web Toolkit (GWT) 1.4

Ajax (Click & Script) Example Script



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see ["Web - HTTP/HTML Protocol" on page 664](#) or go to the [TruClient Help Center](#) (select the relevant version).

VuGen uses the control handler layer to create the effect of an operation on a GUI control. During recording, when encountering one of the supported Ajax controls, VuGen generates a function with an **ajax_xxx** prefix.

In the following example, a user selected item number 1 (index=1) in an Accordion control. VuGen generated an **ajax_accordion** function.

```
web_browser("Accordion.aspx",
            DESCRIPTION,
            ACTION,
            "Navigate=http://labm1app08/AJAX/Accordion/.aspx",
            LAST);
lr_think_time(5);

ajax_accordion("Accordion",
               DESCRIPTION,
               "Framework=atlas",
               "ID=ctl00_SampleContent_MyAccordion",
               ACTION,
               "UserAction=SelectIndex",
               "Index=1",
               LAST);
web_edit_field("free_text_2",
               "Snapshot=t18.inf",
               DESCRIPTION,
               "Type=text",
```

```
"Name=free_text",  
ACTION,  
"SetValue=FILE_PATH",  
LAST);
```

Note: When you record an Ajax session, VuGen generates standard Click & Script functions for objects that are not one of the supported Ajax controls. In the example above, the word FILE_PATH was typed into an edit box.

Ajax (Click & Script) Recording Tips

Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see ["Web - HTTP/HTML Protocol" on page 664](#) or go to the [TruClient Help Center](#) (select the relevant version).

This section lists tips for recording click-and-script Vuser scripts.

Note: Some of the items below apply to specific click-and-script protocols only.

Enable the Functional Testing Agent add-on

If you want to record or edit a click-and-script Vuser script in Internet Explorer, you need to enable the HPE Functional Testing Agent add-on in the browser.

If the browser does not prompt you to enable the add-on, enable it through the Manage Add-ons dialog box (**Tools > Manage add-ons**) in Internet Explorer.

Use the Mouse and not the Keyboard

It is preferable to click on an object with the mouse rather than using the keyboard. During recording, use only GUI objects that are within the browser's pane. Do not use any browser icons, controls, the Stop button, or menu items, such as **View > Refresh**. You may, however, use the Refresh, Home, Back and Forward buttons and the address bar.

Do not Record Over an Existing Script

It is best to record into a newly created script—not an existing one.

Avoid Context Menus

Avoid using context menus during recording. Context menus are right-click menus which pop up when clicking certain objects in a graphical user interface.

Avoid Working in Another Browser While Recording

While recording, do not work in any browser window other than the browser windows opened by VuGen.

Wait for Downloads

Wait for all downloads to complete before doing any action, such as clicking on a button or filling in a text field.

Wait for Pages to Load

During recording, it is best to wait for the page to load completely before doing the next step. If you did not wait for all of the pages to load, record the script again.

Navigate to the Start Page

If the last page in an action does not contain the links and buttons that were available at the start of the iteration, then the next iteration will fail. For example, if the first page has a text link **Book A Flight**, make sure to navigate to the appropriate page at the end of your recording, so that the same link will be visible at the end of the business process.

Use a Higher Event Configuration Level

Record the business process again using the **High** event configuration level. For more information on changing the event configuration level, see ["Click & Script Troubleshooting and Limitations" on page 466](#).

Disable Socket Level Recording

In certain cases, the capturing of the socket level messages disrupts the application. For most recordings, socket level data is not required. To prevent the recording of socket level data, disable the option in the recording options. For more information, see ["GUI Properties > Advanced Recording Options" on page 174](#).

Enable the "Record rendering-related property values" Option

If the client-side scripts of the application use a lot of styling activities, enable the **Record rendering-related property values** option before recording the script. For example, enable this option to record additional DOM properties such as **offsetTop**. Note that enabling this option may decrease the recording speed. You can enable the option by selecting **Recording Options > GUI Properties > Advanced**. For more information, see ["GUI Properties > Advanced Recording Options" on page 174](#).

Ajax (Click & Script) - Replay Tips



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see "[Web - HTTP/HTML Protocol](#)" on page 664 or go to the [TruClient Help Center](#) (select the relevant version).

This section lists tips for replaying click-and-script Vuser scripts.



Note: Some of the items below apply to specific click-and-script protocols only.

Do not re-order statements in a recorded script

Do not change the order of the statements within a recorded script. Also, copying segments of code from one Action to another is not recommended.

Convert non-ASCII characters

If your links contain non-ASCII characters, you should instruct VuGen to convert the data to or from the UTF-8 format.

Enable UTF-8 conversion

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate the **Convert from/to UTF-8** option and set it to **Yes**.

Alternatively, view the list of options that is displayed when a link is not found. Enter the displayed text as-is, such as the hex escape sequences \xA0 or any other non-standard format.

Run the same sequence of actions twice

In some cases, you can perform a certain process only once—such as deleting a user from the database. Replay will fail after the first iteration because the action is no longer valid. Verify that your business process can be repeated more than once with the same data.

Set unique image properties

In the Step Navigator, double click on the previous image step to open its properties. If the **Id**, **Name**, and **Alt** properties are empty, provide further identification of the image, such as its file name in the **Src** property.

Alternatively, you can add an **Ordinal** argument to specify the occurrence number of the image on that page. The **Ordinal** argument uniquely identifies each image on the page where all other identification arguments are not unique. For more information, see the [Function Reference \(Help > Function Reference\)](#).

Check the step's description

If you receive a **GUI Object is not found** error, check the Output pane for a list of the objects in the problematic step. In some cases, the object description changes slightly from run to run.

There are several solutions:

- If the new value is stable, open the script in the Editor and manually modify the value of the step's DESCRIPTION argument.
- If the description changes from run to run, you can use a regular expression in the DESCRIPTION argument. For more information, see the [Function Reference \(Help > Function Reference\)](#).
- Alternatively, replace the problematic object description property, such as Name, with the Ordinal property. For more information, see the [Function Reference \(Help > Function Reference\)](#).

ThreadingModel

Replay of COM script in VuGen fails when the dll registration is missing the **ThreadingModel** string under the **InprocServer32** folder of the GUID.

Ajax (Click & Script) Miscellaneous Tips



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see "[Web - HTTP/HTML Protocol](#)" on page 664 or go to the [TruClient Help Center](#) (select the relevant version).

The following additional tips may help you in troubleshooting problems that you experience with click-and-script Vuser scripts.



Note: Some of the items below apply to specific click-and-script protocols only.

Search for Warnings

Search for warnings or alerts in the Output pane.

Verify the Response

Verify the response of the previous step is correct using **web_reg_find**. For more information, see the Function Reference (**Help > Function Reference**).

Use Alternate Navigation

For problematic steps or those using Java applets, use **Alternative Navigation** to replace the Web step with an HTTP level step. Note that the HTTP level steps may require manual correlations. To perform Alternative Navigation, select a step in the **Step Navigator**, or the text in Script View, and select **Replace with alternative navigation** from the right-click menu.

Working with the Kerberos Protocol

If you are using the Kerberos Protocol for authentication, you must customize VuGen to properly convene authorization sessions. Advanced users can attempt to perform this customization themselves.

In order for the Kerberos Protocol to work properly, create a krb5.ini file and put it in an available folder. Save the full path name of krb5.ini into the KRB5_CONFIG environment variable.

The krb5.ini file should contain detailed information about each domain (KDS and AS addresses) and trust chains.

For more information, contact HPE software support.

Ajax Click & Script Troubleshooting and Limitations

This section describes troubleshooting and limitations for click-and-script protocols.



Caution: Support for the Ajax Click & Script protocol will be discontinued in upcoming versions of LoadRunner and Performance Center.

- Do not use the Ajax Click & Script protocol for new scripts.
- We recommend that you migrate your existing Ajax Click & Script scripts to another Web protocol such as **TruClient - Web** or **Web - HTTP/HTML**.

For details on these Web protocols, see "[Web - HTTP/HTML Protocol](#)" on page 664 or go to the [TruClient Help Center](#) (select the relevant version).



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

Recording Issues and Limitations

Browser support

- Only Internet Explorer is supported for Click & Script. To record browser activity on Firefox, use the Web - HTTP/HTML protocol.
- Not supported for Internet Explorer 10.
- For Click & Script protocols, VuGen may take an excessive amount of time to open the Recording Options dialog box.

Language Support

- Recording an application in a specific language (e.g., French, Japanese) must be performed on a machine whose default locale (in **Settings > Control Panel > Regional Options**) is the same language
- Support of right-to-left languages is limited (e.g., bi-directional or reversed text may not be processed as expected). This is defined by the default operating system translation table.
- The locale of the load-generator machine, must be configured to be the same as that of the recording machine. It cannot be assumed that the Linux default character set is the same as in Windows, even for US-English machines, and this has to be explicitly verified. For example, the default character set on Linux, is UTF-8.

Application behaves differently while being recorded

If your application behaves differently during recording, than it does without recording, you should determine if the recording problem is unique to Web. The effect may be that a Web page will not load, part of the content may be missing, a popup window will not open, and so forth.

Workaround: Create a new Web - HTTP/HTML script and repeat the recording.

In the event that the recording fails in Web - HTTP/HTML, we recommend that you disable socket level recording (see ["Click & Script Recording Tips" on page 459](#)).

The problem may be the result of an event listener. Use trial and error to disable event listeners in the **Web Event Configuration** Recording Options, and then re-record your session as a Web - HTTP/HTML user.

Certain Click & Script steps do not generate properly

After recording a script, if not all steps are correctly generated, the problem may be due to the **Windows Component > Internet Explorer Enhanced Security Configuration**.

Remove **Internet Explorer Enhanced Security Configuration** by selecting **Control Panel > Add or Remove Programs > Add or Remove Windows Components** and re-record your script.

Disable an Event Listener

1. Click **Record > Recording Options** to open the Recording Options dialog box.
2. Select the **GUI Properties > Web Event Configuration** node.
3. Click **Custom Settings** and expand the **Web Objects** node. Select an object.
4. Select **Disabled** from the list in the **Record** column for the relevant Web object. If the recording still does not work, enable the listener you previously disabled, and try disabling another one. Repeat these steps until your recording succeeds.

Dynamic menu navigation was not recorded

A dynamic menu is a menu that dynamically changes depending on where you select it. If the dynamic menu navigation was not recorded, record again using "high" event configuration mode. These settings can be found in the **Recording Options > GUI Properties > Web Event Configuration** node.

Certain user actions were not recorded

Check if there is a Java applet running inside the browser. If not, record the script with the Web - HTTP/HTML protocol.

Replay Issues

GUI object not found

Does the error occur at the beginning of the second iteration?

If the error occurs at the beginning of the second iteration's Action section, it is probably the result of a starting page that was present for the first iteration, but missing for the second one. If the last page in an action does not contain the links and buttons that were available at the start of the iteration, then the next iteration will fail. For example, if the first page has a text link **Book A Flight**, make sure to navigate to the appropriate page, so that the same link will be visible at the end of the business process.

Is it a text link containing non-ASCII characters?

If the problem occurs with non-ASCII characters, you should instruct VuGen to convert the data to a suitable character set.

Enable Data Conversion on Windows Machines

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate **Charset Conversions by HTTP** in the Web (Click & Script) > General options, and set it to **Yes**.

Enable UTF-8 conversion for Linux Machines

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the **Advanced Options** dialog box.
3. Locate **Convert from/to UTF-8** in the General options and set it to **Yes**

Alternatively, view the list of alternatives that are displayed when a link is not found. Enter the displayed text as-is, such as hex escape sequences \xA0 or any other non-standard format.

Can you run the same sequence of actions twice in the application?

In some cases, you can only perform a certain process once, such as deleting a user from the database. Replay will fail after the first iteration, because the action is no longer valid. Verify that your business process can be repeated in the application more than once with the same data, without recording again.

Were the image properties 'Id', 'Name' and 'Alt' empty?

In the **Step Navigator**, double click on the previous image step to open its properties. If the **Id**, **Name**, and **Alt** properties are empty, provide further identification of the image, such as its file name in the **Src**

property.

Alternatively, you add an **Ordinal** argument to specify the occurrence number of the image on that page. The **Ordinal** argument uniquely identifies each image on the page where all other identification arguments are not unique. For more information, see the Function Reference (**Help > Function Reference**).

Did the step's description change?

Check the Output pane for a list of the objects in the problematic step. In some cases, the object description changes slightly from run to run.

There are several solutions:

- If the new value is stable, open the Script View and manually modify the value of the step's DESCRIPTION argument(s).
- If the description changes from run to run, you can use a regular expression in the DESCRIPTION argument(s). For more information, see the Function Reference (**Help > Function Reference**).
- Alternatively, replace the problematic object description property, such as Name, with the Ordinal property. For more information, see the Function Reference (**Help > Function Reference**).

Did the page load completely during recording?

During recording, it is best to wait for the page to load completely before doing the next step. If you did not wait for all of the pages to load, record the script again.

Replay failure

If the replay fails at a particular step, check the step description. VuGen may have interpreted a single space as a double space. Make sure that there are no incorrect double spaces in the string.

Replay snapshots

Replay snapshots may differ from the actual Web page.

Memory Issues

Out of memory error in JavaScript

Increase the JavaScript memory in the runtime settings.

Increase the JavaScript Memory Size

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate the **Memory Management JavaScript Runtime Memory Size (Kb)** and **Memory Management JavaScript Stack Memory Size (Kb)** options.
4. Increase the memory sizes to 512Kb or higher.

VuGen displays JavaScript errors

If VuGen displays JavaScript errors in the Output pane, enable IE (Internet Explorer) script errors in order to verify that the Javascript itself does not contain errors.

Show Script Errors

1. Open Internet Explorer.
2. Select **Tools > Internet Options** and click the **Advanced** tab.
3. Under **Browsing**, select the **Display a notification about every script error** check box.
4. Rerun the application in IE. If IE displays script errors, then there is a problem with the JavaScript application. If it is not possible to fix the application, you can safely ignore the corresponding replay errors.

Problems following parameterization

If you encounter problems only after you have parameterized values, verify that the values are valid for your application. Perform business process with the value of the parameter and verify that the application accepts it.

Problems with applications that utilize styling actions

If the client-side scripts of the application use a lot of styling activities, you should record the script again after enabling the **Record rendering-related property values** option. This enables the recording of additional DOM objects.

Enable the "Record rendering-related property values" Option

1. Select **Recording > Recording Options** and select the **GUI Properties > Advanced** node.
2. Select the **Record rendering-related property values** check box.
3. Re-record the Vuser script.

Supported Environments

- ActiveX objects and Java applets are only supported on Windows platforms.
- Not supported for Macromedia Flash or VB Script.
- Click & Script protocols do not support pop-up windows.

Citrix Protocol

Citrix Protocol - Overview

Citrix Vuser scripts emulate Citrix ICA protocol communication between a Citrix client and server. VuGen records all activity during the communication and creates a Vuser script.

Before you can create scripts for the Citrix protocol, you must set up and configure your Citrix environment to work with VuGen.

After the system is properly configured, VuGen generates Citrix functions when you perform actions on the remote server. Each function begins with a **ctrx** prefix. These functions emulate the analog movements of the mouse and keyboard. For example:

```
ctrx_mouse_click(44, 318, LEFT_BUTTON, 0, CTRX_LAST);
```

In addition, you can use other **ctrx** functions to synchronize the replay of the actions, by waiting for specific windows to open.

VuGen also allows you to record a Citrix Web interface session. In this case, the Citrix client is installed, but your interface is a browser instead of a client interface. To record Citrix Web interface sessions, you must perform a multi-protocol recording for Citrix and Web Vusers. In multi-protocol mode, VuGen generates functions from both Citrix and Web protocols during recording.



See also:

- For more information about the syntax and parameters, see the [Function Reference \(Help > Function Reference\)](#).

Set Up Your Citrix Environment

Before creating a Citrix script, you need to properly configure your Citrix client and server.

1. **Prerequisite:** Ensure that you are working with supported versions of your Citrix client and server as listed in the [System Requirements](#).
2. Set up your Citrix ICA clients.
 - a. **Install the client on the VuGen and Load Generator machines.** To run your script, you must install a Citrix client on the VuGen machine each Load Generator machine. If you do not have a client installed, you can download one from the Citrix website www.citrix.com under the **Downloads** section.
 - b. **Verify the required VuGen Citrix settings.** For some Citrix versions, the **TCP/IP** Citrix connection option does not work. In that case, use the **TCP/IP+HTTP** Citrix connection option: In the VuGen Recording Options dialog box, select **Citrix > Login**. In the **Connection** area, select **TCP/IP+HTTP** as the **Network Protocol**. For more details, see "[Citrix > Login Recording Options](#)" on page 142.
3. Set up your Citrix Web Interface clients. (Only when connecting to Citrix server via Web Interface)
 - a. **DEP.** Before recording, Turn off Microsoft DEP on the VuGen machine. This is done in different ways on different operating systems.
One way to do this on Vista, Windows 7, and Windows 2008 is as follows:
 - i. From the cmd shell, run **bcdedit.exe/set nx AlwaysOff**.
 - ii. Restart the computer.
 - b. **Security Software.** If possible, disable anti-malware and other security or antivirus software. Alternatively, add an exception to ignore **vugen.exe**, **runcitrixclient.exe** (a VuGen process) and **wfica32.exe** (the Citrix client process). Otherwise, your security software may suspect VuGen's

Web recording engine and prevent VuGen from launching your application.

- c. **Disable the desktop toolbar.** The Citrix administrator should disable the desktop toolbar. There are several ways to do this. Add `ConnectionBar=0` to the **default.ica** file, or follow the method described in: <https://support.citrix.com/article/CTX138928>. (Relevant only for published desktops, not published applications)
- d. **Internet Explorer Settings.**
 - i. When working with Internet Explorer 8 or later, ensure that the **Enable SmartScreen Filter** option (**Internet Options > Advanced**) is not selected.
 - ii. When recording on the Citrix Web Interface from a Windows server operating system, **Internet Explorer Enhanced Security** (IE ESC) must be disabled. This option is set in different locations in different operating systems.

For example, in Windows 2008 server, you set this option from the Server Manager, using the Configure IE ESC option.

- e. **VuGen Settings.**
 - i. Make sure that VuGen is set to create scripts containing only explicit URLs.

In the VuGen Recording Options dialog box, select **General > Recording** and click the HTML Advanced button. In the Script type area, select: **A script containing explicit URLs only.**
 - ii. Enable the Citrix server SessionToken correlation rules:

In the VuGen Recording Options dialog box, select **Correlations > Rules**, expand the **Citrix_XenApp** node and select the relevant token option for the Citrix version you are using. (If you are not sure which option to select, you can select all of them.)

- 4. Set your Citrix client to work in non-seamless mode.

VuGen can create Citrix scripts only in non-seamless mode (client opens within a Citrix ICA window and not as a local application).

In the Citrix client, select **Preferences > Session Settings** and set **Window size** to **No preference**.

This setting is saved across Citrix settings for the same user. Alternatively, add `TWIMode=Off` in the **default.ica** file on the server.

- 5. Configure the Citrix server.
 - a. **Session Disconnect.** By default, when a client times out or disconnects from the Citrix server, the session remains open for a defined time period. However, beginning a run in a Citrix session that has an unpredictable state can cause your test to fail.

Therefore, the Citrix server administrator should configure the Citrix server to end (reset) the client session when a client disconnects for any reason.
 - b. **Multi-Session Support.** If you are going to run more than one Citrix Vuser on a load generator, ensure that the Citrix server is configured to enable multiple sessions per user/client.

- 6. Install the VuGen Citrix agent on the Citrix server. (Optional)

The VuGen Citrix Agent provides an additional level of object-oriented information about the Citrix published application or desktop that enables you to improve the functionality and robustness of

your Citrix script. It also provides some extra enhancements over the normal Citrix functionality. For details, see ["Agent for Citrix Presentation Server - Overview" below](#).

- a. If you are upgrading the agent, make sure to uninstall the previous version before installing the next one.
- b. If your Citrix server requires administrator permissions to install software, log in as an administrator to the server.
- c. Ensure that Microsoft DEP is fully disabled. (DEP must be disabled when using the agent.)
- d. Locate the installation file, **SetupCitrixAgent.exe**, on the HPE product installation disk in the **Additional Components\Agent for Citrix Server** folder.
- e. Follow the installation wizard to completion.
- f. To enable logging for the agent, set **DebugEnabled=1** in the [General1] section of CtrxAgent.ini file. The log file will be created in either the Agent's bin\ folder or in session's %TEMP% folder, depending on user permissions.

Note:

- After installation, the agent will be active only for LoadRunner invoked Citrix sessions—it will not be active Citrix sessions started without LoadRunner.
- To uninstall, select **HPE Software Agent for Citrix Presentation Server** in the Add/Remove Programs window on the server machine.
- The Citrix agent was designed to be cross-compatible, allowing you to use one version of LoadRunner and yet another version of the agent. It is recommended that you use the most recent version of the agent.

Agent for Citrix Presentation Server - Overview

The Agent for Citrix Presentation Server, or Citrix Agent, is an optional utility that you can install on the Citrix server. It provides enhancements to the normal Citrix functionality. The following sections describe these enhancements.

It is provided in the product's installation disk and you can install it on any supported Citrix server. For more information, see ["Install the VuGen Citrix agent on the Citrix server. \(Optional\)" on the previous page](#) For details on supported Citrix server versions, see the [System Requirements](#).

Object Detail Recording

If the Citrix Agent is installed on the Citrix server, then when you record a Citrix script, VuGen records specific information about the active object instead of general information about the action. For example, VuGen generates **Obj Mouse Click** and **Obj Mouse Double Click** steps instead of the **Mouse Click** and **Mouse Double Click** that it generates without the agent.

The following example shows the same mouse-click action recorded with and without the agent installation. Note that with an agent, VuGen generates `ctrx_obj_xxx` functions for all of the mouse actions, such as click, double-click, and release.

```
/* Without Agent Installation */  
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0, test3.txt - Notepad);  
/* With Agent Installation */  
ctrx_obj_mouse_click("<text=test3.txt - Notepad class=Notepad>" 573,  
    61, LEFT_BUTTON, 0, test3.txt - Notepad=snapshot21, CTRX_LAST);
```

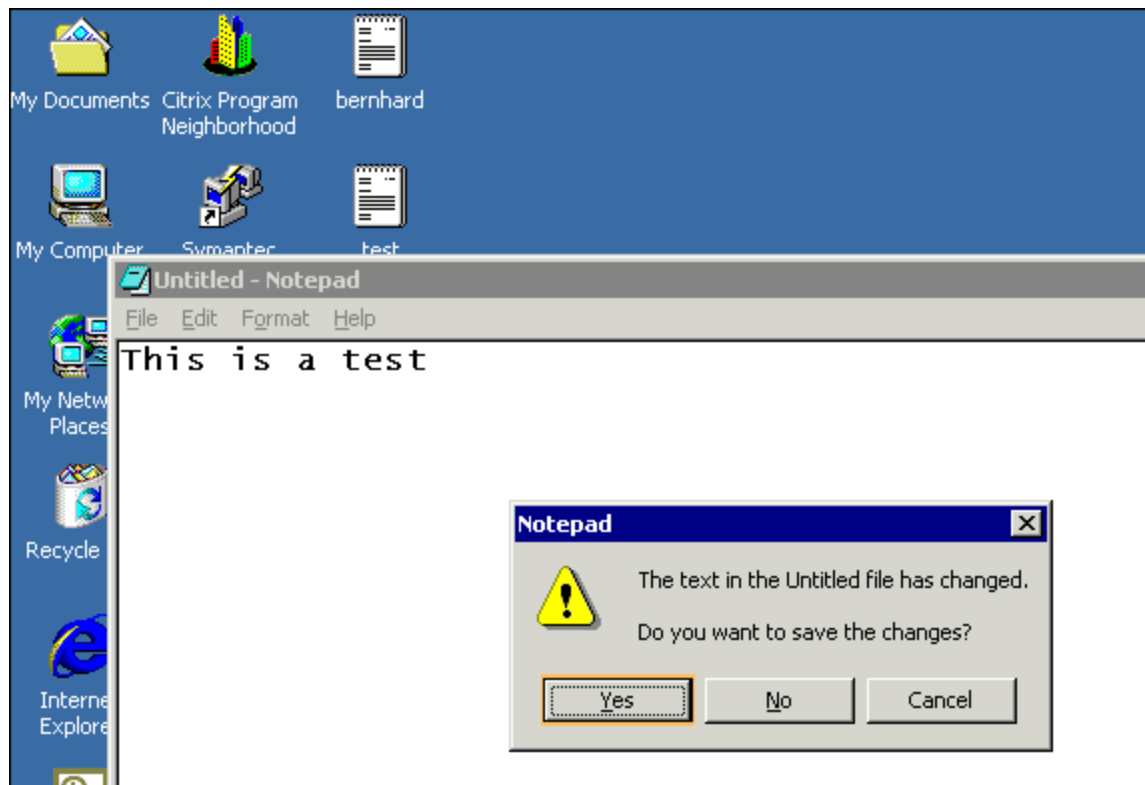
In the example above, the first argument of the **ctrx_obj_mouse_click** function contains the text of the window's title and the class, Notepad. Note that although the agent provides additional information about each object, Vusers only access objects by their window name and the object coordinates.

Active Object Recognition

The Citrix Agent lets you see which objects VuGen detects in the client window. This includes all Windows Basic Objects such as edit boxes, buttons, and item lists in the current window.

To see the detected objects, you move your mouse through the snapshot. VuGen highlights the borders of the detected objects as the mouse passes over them.

In the following example, the **Yes** button is one of the detected objects.



Expanded Right-Click Menu

When you click within a snapshot, you can insert several functions into the script using the right-click menu. When no agent is installed, you are limited to the **Insert Mouse Click**, **Insert Mouse Double Click**, **Insert Sync on Bitmap**, and **Insert Get Bitmap Value**. If you are using a 256-color set, the **Insert Sync on Bitmap** and **Get Bitmap Value** steps are not available from the right-click menu.

If the Citrix Agent is installed on the Citrix server, the following additional options are available from the right-click menu of the Citrix window in focus:

- **Obj Get Info** and **Sync on Obj Info**. Provide information about the state of the object: ENABLED, FOCUSED, VISIBLE, TEXT, CHECKED, and LINES.
- **Insert Sync on Obj Info**. Instructs VuGen to wait for a certain state before continuing. This is generated as a **ctrx_sync_on_obj_info** function.
- **Insert Obj Get Info**. Retrieves the current state of any object property. This is generated as a **ctrx_get_obj_info** function.
- **Insert Sync on Text** and **Get Text**.

For more information on the above-mentioned synchronization steps, see ["Citrix - Automatic Synchronization" on page 444](#) and ["Citrix - Manual Synchronization" on page 446](#).

These commands are interactive—when you insert them into the script, you are prompted to mark the object or text area in the snapshot.

In the following example, the **ctrx_sync_on_obj_info** function provides synchronization by waiting for the Font dialog box to come into focus.

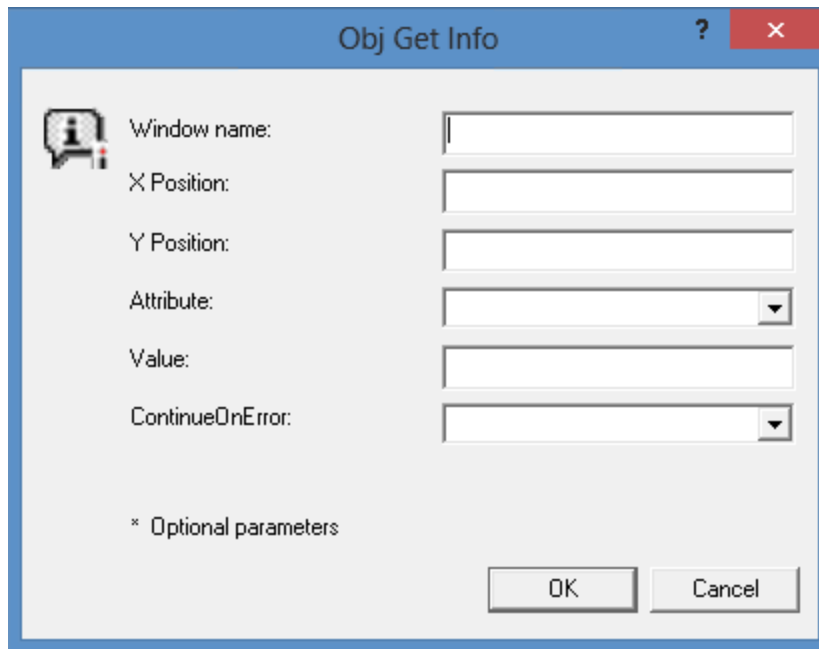
```
ctrx_sync_on_obj_info("Font", 31, 59, FOCUSED, "TRUE", CTRX_LAST);
```

Utilizing VuGen's ability to detect objects, you can perform actions on specific objects interactively, from within the snapshot.

Insert a Function Interactively Using the Agent Capabilities

1. Click at a point within the Step Navigator to insert the new step. Make sure that a snapshot is visible.
2. Click within the snapshot.
3. To mark a bitmap:
 - a. Select it and choose **Insert Sync on Bitmap** from the right-click menu.
 - b. Mark the required area.
 - c. Click **OK** in the Sync on Bitmap dialog box. VuGen inserts the step into the script after the currently selected step.
4. For all other steps, move your mouse over snapshot objects to determine which items are active—VuGen highlights the borders of active objects as the mouse passes over them.

When the object is highlighted, right-click and select one of the Insert commands. A dialog box opens with the step's properties.



Set the desired properties and click **OK**. VuGen inserts the step into your script.

Text Retrieval

With the agent installed, VuGen lets you save standard text to a buffer. Note that VuGen can only save true text—not a graphical representation of text in the form of an image.

You save the text using the **Get Text** step either during or after recording.

For additional details, see ["Citrix - Manual Synchronization" on page 446](#).

Citrix Recording Tips

Before recording a Citrix Vuser script, make sure your environment is set up as described in ["Set Up Your Citrix Environment" on page 436](#), and then consider these guidelines:

Plan before recording

Make sure you have a well-defined business process planned, and run through it before recording it in VuGen.

Avoid production environments if possible

Try to load test Citrix applications which are restricted (contained) to a few Citrix servers in a Citrix development/test “farm” rather than to load test in a live Citrix production environment.

Consider creating your first test using single-protocol script before testing the Web interface

Consider using a single-protocol Citrix ICA script to load test the ICA elements of your application before expanding the test to address the Web interface. Once you have stabilized the ICA steps of your

script, it may be easier to troubleshoot any problems that arise in your multi-protocol script.

Record into appropriate sections

Record the connection process into the **vuser_init** section, and the closing process into the **vuser_end** section. This will prevent you from performing iterations on the connecting and disconnecting. For more information about recording into sections, see ["Vuser Script Sections" on page 132](#).

Ensure a clean session

When recording a session, make sure to perform the complete business process, starting with the connection and ending with the cleanup. End your session at a point from where you could start the entire process from the beginning. Do not leave any client or application windows open.

Use explicit clicks

When opening expanded menu options, click explicitly on each option—do not depend on the expanding menu. For example, when choosing **Start > All Programs > Microsoft Word**, be sure to click on the line **All Programs**.

Do not resize windows

To ensure exact reproduction of recorded actions, avoid moving or resizing windows while recording. If it is absolutely necessary to change the size or position of a window, double-click on the relevant **Sync on Window** step in the **Step Navigator** and modify the window's coordinates. This will often, but not always give the desired replay results.

Make sure resolution settings are consistent

To ensure successful bitmap synchronization, make sure that the resolution settings match. On the recording machine, check the settings of the Citrix client, the Recording Options, and the runtime settings. On the load generators, check the settings of the Citrix client, and make sure that they are consistent between all load generators and recording machines. If there is an inconsistency between the resolutions, the server traffic increases in order to make the necessary adjustments.

Add Manual Synchronization Points

If it is necessary to wait for an event during recording, such as the opening of an application, add manual synchronization points, such as **Sync on Text** (if using the Citrix agent) or **Sync on Bitmap**. For details, see ["Citrix - Manual Synchronization" on page 446](#).

Use Classic Windows Style

For **Sync on Bitmap** steps, record windows in the "classic" windows style—not the XP style. To do this:

1. Click in the desktop area.
2. Select **Properties** from the right-click menu.
3. Click the Theme tab.
4. From the **Theme** list, select **Windows Classic**.
5. Click **OK**.

Modify or disable DEP

DEP (Data Execution Prevention) is a security feature included in Windows. It can interfere with some of the Citrix Agent's functionality during record and replay, and may cause Internet Explorer to hang on the VuGen machine. If you experience unusual behavior during recording, modify the DEP settings.

1. Open **Start > Control Panel > System and Security > System > Advanced system settings**. The System Properties dialog box opens.
2. Select the **Advanced** tab, click the **Settings** button in the **Performance** section.
3. In the Performance Options dialog box, click the **Data Execution Prevention** tab. Select the first option, **Turn on DEP for essential Windows programs and services only**.

If you cannot change this option, click **Add**. Browse to the client program, for example IEXPLORE.EXE and click **Open** to add the application to the exception list.

4. If neither of the above options are possible, try to disable DEP completely.
 - a. Open a command prompt.
 - b. Run the following command: **bcdedit.exe /set {current} nx AlwaysOff**
 - c. Reboot the machine.
 - d. Verify that the settings took effect by running the following at the command line: **BCDEdit /enum**
 - e. Verify that the last line shows **nx AlwaysOff**.

Disable Active X

If the application containing the Citrix session is a native Citrix binary file, disable ActiveX controls in order to prevent execution of the Citrix Online plugin. To disable ActiveX, in Internet Explorer go to **Tools > Internet Options > Security** tab. Click **Custom level** and under **Run ActiveX controls and plug-ins** select **Disable**. Make sure that ActiveX is disabled for the Internet Zone to which the Citrix WebInterface site belongs.

If you cannot change above settings, try to suppress the loading of the Citrix ActiveX object into the browser's process by setting the **SuppressCitrixOcxEnabledto** flag to true in **LR\config\bbhook.ini**.

Disable Client Updates

Disable client updates when prompted by the Citrix client. This will prevent forward compatibility issues between VuGen and newer Citrix clients that are not currently supported. (For details on supported versions, see the [System Requirements](#).)

For more help, see "[Citrix - Troubleshooting and Limitations](#)" on page 452.

Citrix Synchronization

Synchronization refers to waiting for windows and objects to become available before executing an action. This is necessary when recording Citrix scripts because, for example, if a step in a script opens a window, and the next step performs an action in that window, the second step cannot be implemented until the window opens. In order to ensure that VuGen does not replay the script incorrectly, it

automatically generates functions that synchronize the script by waiting for windows or objects to become available. In addition, you can add synchronization functions manually.

To choose the most suitable synchronization mechanisms for a particular use-case scenario, it is crucial to understand the benefits and drawbacks of the different synchronization mechanisms:

- **Bitmap synchronization (Exact level).** The fastest, most reliable, and least consuming synchronization operation. Should be used by default.
- **Bitmap synchronization (non-Exact levels).** Consumes additional resources of the load generator machine. In general, this mechanism is not recommended unless you specifically know that you need it for a particular purpose.
- **_obj_ sync functions.** Might be a reasonably good way to improve script stability but requires Agent presence on the server side, which cannot always be achieved. Also creates additional traffic to the server (this should not be an issue but should be at least considered).
- **_text.** Same as _obj_. Might not be supported for some applications or technologies.
- **_ocr.** Use with caution! Consumes significant amounts of hardware resources on the load generator machine.

Exact-level bitmap synchronization should cover most use-case scenarios. Other synchronization mechanism usage should be properly motivated.

For more information about **_text** and **_ocr** functions, see the relevant [Synchronization Functions](#) in the Function Reference.



See also:

- ["Citrix - Automatic Synchronization" below](#)
- ["Citrix - Manual Synchronization" on page 446](#)

Citrix - Automatic Synchronization

During recording, VuGen automatically generates steps that help synchronize the Vuser's replay of the script.

Sync on Window

The **Sync on Window** step instructs the Vuser to wait for a specific event before resuming replay. The available events are **Create** and **Active**. The Create event waits until the window is created. The Active event waits until the window is created and then activated (in focus). Usually VuGen generates a function with a Create event. If, however, the next instruction is a keyboard event, VuGen generates a function with an Active event.

In the Editor, the corresponding function call to the **Sync on Window** step is **ctrl_sync_on_window**.

Sync on Obj Info

The **Sync on Obj Info** step instructs the Vuser to wait for a specific object property before resuming replay. The available attributes are **Enabled**, **Visible**, **Focused**, **Text**, **Checked**, **Lines**, or **Item**. The

Enabled, Visible, Focused, and Checked attributes are boolean values that can receive the values **true** or **false**. The other attributes require a textual or numerical object value.

A primary objective of this step is to wait for an object to be in focus before performing an action upon it.

VuGen automatically generates **sync_on_obj_info** steps when the Citrix agent is installed and the Use Citrix Agent Input in Code Generation option is enabled in the Recording Options. By default, this Recording option is enabled. For more information, see ["Citrix > Code Generation Recording Options" on page 142](#).

```
ctrx_sync_on_obj_info("Run=snapshot9", 120, 144, TEXT, "OK",  
                     CTRX_LAST);
```

Sync on Text

A text synchronization step, **Sync on Text**, instructs the Vuser to wait for a text string to appear at the specified position before continuing. When replaying **Sync on Text**, the Vuser searches for the text in the rectangle whose modifiable coordinates are specified in the step's properties.

Note:

- The maximum allowable length of a text string is 255 characters.
- The Citrix agent does not support this function on Windows 10, Windows Server 2016, or later versions of Windows.

By default, automatic text synchronization is disabled. However, with the Citrix Agent installed (see ["Agent for Citrix Presentation Server - Overview" on page 438](#)), you can instruct VuGen to automatically generate a text synchronization step before each mouse click or double-click. After this option is enabled, it applies both to newly recorded scripts and also to existing scripts if you choose the **Record > Regenerate Script** option. For more information, see ["Citrix > Code Generation Recording Options" on page 142](#).

In the Editor, the corresponding function call to the **Sync on Text** step is **ctrx_sync_on_text_ex**.

The following segment shows a **ctrx_sync_on_text_ex** function that was recorded during a Citrix recording with the Citrix Agent installed and text synchronization enabled.

```
ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0, 0, 391, 224, "snapshot1",  
CTRX_LAST);  
ctrx_sync_on_text_ex (196, 198, 44, 14, "OK", "ICA Seamless Host Agent=snapshot2",  
CTRX_LAST);  
ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198, LEFT_BUTTON, 0, "ICA  
Seamless Host Agent=snapshot2", CTRX_LAST);
```

For more information on this function, see the Function Reference (**Help > Function Reference**).

See ["Citrix - Additional Ways to Synchronize Your Script" on the next page](#) for additional information.

Citrix - Manual Synchronization



You can manually add synchronization steps both during and after recording. A common use of this capability is where the actual window did not change, but an object within the window changed. Since the window did not change, VuGen did not detect or record a **Sync on Window**.

For example, if you want the replay to wait for a specific graphic image in a browser window, you insert manual synchronization. Or, if you are recording a large window with several tabs, you can insert a synchronization step to wait for the new tab's content to open.

Note: You can also instruct LoadRunner to insert automatic synchronization steps during your recording session. For details, see ["Citrix - Automatic Synchronization" on page 444](#).

Synchronize manually during recording

To add synchronization during recording, you use the floating toolbar. The **Sync on Bitmap** and **Sync on Text** functions enable you to mark an area or text that needs to be visible within the client window before resuming replay.

- To insert a **Sync on Bitmap** step, click the **Insert Sync on Bitmap**  button on the toolbar and mark a rectangle around the desired area.
- To insert a **Sync on Text** step (Citrix Agent required), click the **Insert Sync on Text**  button on the toolbar and mark a rectangle around the desired text.

Synchronize manually after recording

You can also add synchronization after the recording session. To add a synchronization step, right-click in the Snapshot pane, and select a synchronization option:

- **Insert Sync on Bitmap.** Replay waits until the marked bitmap appears. You mark a rectangle around the desired area.
- **Insert Sync on Bitmap (by coordinates).** This option is identical to the one above, except that it allows you to manually enter coordinates for the synchronization area.
- **Insert Sync on Obj Info.** Waits until an object's attributes have the specified values (agent installations only).
- **Sync on Text.** Waits until the specified text is displayed (agent installations only).

Citrix - Additional Ways to Synchronize Your Script

In addition to [automatic](#) and [manual](#) synchronization in Citrix Vuser scripts, you can add certain other steps that affect the synchronization indirectly:

Setting a Delay

The **ctrx_set_waiting_time** step sets a waiting time for the other Citrix synchronization functions. This setting applies to all functions that follow it within the script. For example, if your **ctrx_sync_on_window**

steps are timing out, you can increase the default timeout from 60 seconds to 180 seconds.

To insert this step, insert a **ctrx_set_waiting_time** step from the **Steps** toolbox.

Checking if a Window Exists or Closed

The **ctrx_win_exist** step checks if a window is visible in the Citrix client. By adding control flow statements, you can use this function to check for a window that does not always open, such as a warning dialog box. In the following example, **ctrx_win_exist** checks whether a browser was launched. The second argument indicates how long to wait for the browser window to open. If the window did not open in the specified time, it double-clicks the application's icon to open it.

```
if (!ctrx_win_exist("Welcome",6, CTRX_LAST))  
    ctrx_mouse_double_click(34, 325, LEFT_BUTTON, 0, CTRX_LAST)
```

To insert this step, insert a **ctrx_win_exist** step from the **Steps** toolbox.

Another useful application for this step is to check if a window has been closed. If you need to wait for a window to close, you should use a synchronization step such as **ctrx_unset_window**.

For detailed information about these functions, see the Function Reference (**Help > Function Reference**).

Waiting for a Bitmap Change

In certain cases, you do not know what data or image will be displayed in an area, but you do expect it to change. To emulate this, you can use the **ctrx_sync_on_bitmap_change** function. Right-click in the snapshot, and select **Insert Sync on Bitmap** from the right-click menu. VuGen inserts the step or function at the location of the cursor.

The syntax of the functions is as follows:



Example:

```
ctrx_sync_on_bitmap (x_start, y_start, width, height, hash, CTRX_LAST);  
ctrx_sync_on_bitmap_change (x_start, y_start, width, height, [initial_wait_time,]  
[timeout,] [initial_bitmap_value,] CTRX_LAST);
```

The following optional arguments are available for **ctrx_sync_on_bitmap_change**:

- **initial_wait_time**. When to begin checking for a change.
- **timeout**. The amount of time in seconds to wait for a change to occur before failing.
- **initial_bitmap_value**. The initial hash value of the bitmap. Users wait until the hash value is different from the specified initial bitmap value.

In the following example, the recorded function was modified and assigned an initial waiting time of 300 seconds and a timeout of 400 seconds.

```
ctrx_sync_on_bitmap_change(93, 227, 78, 52,  
300,400, "66de3122a58baade89e63698d1c0d5dfa",CTRX_LAST);
```

Note: If you are using **ctrl_sync_on_bitmap**, make sure that the screen settings in the Controller, Load Generator machine, and VuGen are the same. Otherwise, VuGen may be unable to find the correct bitmaps during replay. For information on how to configure the client settings, see ["Recording Options" on page 141](#).

Failed Bitmap Synchronization Dialog Box

This dialog box enables you to decide what to do when a bitmap synchronization fails.

To access	This dialog box opens automatically during replay when there is a mismatch between the record snapshot and the replay snapshot in a bitmap synchronization step.
------------------	--

User interface elements are described below:

UI Element	Description
Continue	Accept the mismatch and use both the original and new snapshots as a basis for comparison between screens during future replays. If replay returns either one of the bitmaps, the Vuser will not fail.
Recording Snapshot	A view of the recording snapshot.
Replay Snapshot	A view of the replay snapshot.
Stop	Consider the mismatch between the snapshots to be an error. This error will be handled like all other errors and halt the execution by default. Alternatively, you can specify Continue on Error for a specific function as described in "Citrix - Troubleshooting and Limitations" on page 452 .

Citrix Replaying Tips

Before replaying a Citrix Vuser script, consider these guidelines:

Wildcards

You can use wildcards (*) in defining window names. This is especially useful where the window name may change during replay, by its suffix or prefix.

In the following example, the title of the **Microsoft Internet Explorer** window was modified with a wildcard.

```
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0,  
"Welcome to MSN.com - Microsoft Internet Explorer");ctrx_mouse_click(573, 61, LEFT_  
BUTTON, 0,  
"* - Microsoft Internet Explorer");
```

For more information, see the Function Reference (**Help > Function Reference**).

Run as a Process—not a Service

Since the Citrix protocol is GUI-based, it relies on several settings which are imperative for enabling interactivity. The VuGen script was recorded in an interactive session configured with specific screen settings, ClearType options, keyboard layouts, and so forth. If you run the test as a service, which by default, uses the SYSTEM account, the settings will most likely be different. Any mismatch in the above settings may result in a failed replay. Therefore, you should run the test as a process.

Modify the window size of the Citrix client during replay

Note: Changing the Citrix client window size for replay changes the resolution as well. If your script contains synchronization calls, this may introduce synchronization errors during replay.

1. Close VuGen.
2. In the Citrix script folder, open **default.cfg**.
3. Under the **[CITRIX]** section, modify the following line:

```
Window=<width> x <height>
```

where **width** and **height** are defined in pixels.

For example:

```
Window=1024 x 768
```

4. Save the file.

Enable Think Time

For best results, ensure that think time is enabled in the runtime settings. Think time is especially relevant before the **ctrx_sync_on_window** and **ctrx_sync_on_bitmap** functions, which require time to stabilize.

Set Initialization Quota

To prevent overloading by multiple Vusers while connecting, set an initialization quota of 4 to 10 Vusers (depending on the capacity of the server) or apply ramp-up initialization using the Scheduler.

Set a Bitmap Polling Delay

To prevent a false failure in bitmap synchronization, set the **Bitmap polling delay** in the runtime settings **Citrix > Synchronization** view, to a non-zero value. A recommended value is 1000 msec.

Use Exact Tolerance

It is recommended to always use **Exact** tolerance for synchronization. Set the **Default Image Sync Tolerance** in the runtime settings **Citrix > Synchronization** view, to **Exact**. Setting this option to **Non-exact** is not effective for slight changes, such as a difference in the **ClearType** settings.

Set Consistency Between Machines

If you intend to replay the script on another machine, make sure that the following items are consistent between the record and replay machines: Window Size (resolution), Window Colors, System Font, ClearType, and the other Default Options settings for the Citrix client. These settings affect the hash value of bitmaps, and inconsistencies may cause replay to fail. To view the Citrix client settings, right-click an item from the Citrix program group and select **Application Set Settings** or **Custom Connection Settings**. (Note that the remote session on the Citrix server inherits the ClearType settings of the local machine.)

Increasing the Number of Vusers per Load Generator Machine

Load Generator machines running Citrix Vusers may be limited in the number of Vusers that can run, due to the graphic resources available to that machine, also known as the GDI. To increase the number of Vusers per machine, you can open a terminal server session on the machine to act as an additional load generator.

The GDI count is operating system-dependent. For example, the actual GDI (Graphics Device Interface) count for a heavily loaded machine using LoadRunner is approximately 7,500. The maximum available GDI on most Windows operating systems is 16,384.

For more information on creating a terminal server session, see [Terminal Services Overview](#).

Note: By default, sessions on a terminal server use a 256-color set. If you intend to use a terminal session for load testing, make sure to record on machines with a 256-color set.

Citrix Debugging Tips

When your test does not run as expected, try these debugging tips:

Single Client Installation

If you are unsuccessful in recording any Citrix actions, verify that you have only one Citrix client installed on your machine. To verify that only one client is installed, open the Add/Remove Programs dialog box from the Control Panel and make sure that there is only one entry for a Citrix client.

Add Breakpoints

Add breakpoints to your script in VuGen to help you determine the problematic lines of code.

Synchronize Your Script

If replay fails, you may need to insert synchronization functions into your script to allow more time for the desired windows to come into focus. Although you can manually add a delay using **lr_think_time**, we recommend that you use one of the synchronization functions discussed in ["Citrix - Automatic Synchronization" on page 444](#).

Regenerate Your Script

During recording, VuGen saves all of the agent information together with the script. By default, it also includes this information in the script, except for the **ctrx_sync_on_text** steps. If you encounter text synchronization issues, regenerate the script to include the text synchronization steps.

In addition, if you disabled the generation of agent information in the recording options, you can regenerate the script to include them.

To regenerate a script, select **Record > Regenerate Script** and select the desired options. For more information about regenerating scripts, see ["Regenerate a Vuser Script" on page 220](#).

Continue on Error

You can instruct Vusers to continue running even after encountering an error, such as not locating a matching window. You specify Continue on Error for individual steps.

This is especially useful where you know that one of two windows may open, but you are unsure of which. Both windows are legal, but only one will open.

To indicate Continue on Error:

1. In the **Step Navigator**, right-click on the step and select **Show Arguments**. In the **Continue on Error** box, select the **CONTINUE_ON_ERROR** option.
2. In **Script view**, locate the function and add **CONTINUE_ON_ERROR** as a final argument, before **CTRX_LAST**.

This option is not available for the following functions:

- **ctrx_key**
- **ctrx_key_down**
- **ctrx_key_up**
- **ctrx_type**
- **ctrx_set_waiting_time**
- **ctrx_save_bitmap**
- **ctrx_execute_on_window**
- **ctrx_set_exception**

View the Extended Log

You can view additional replay information such as:

- the total amount of GDI handles being used
- a list of the running Citrix processes (concentr.exe, receiver.exe, wfica32.exe, wfcrun32.exe) with their PIDs and user names (if the process is not running under the LoadRunner user)
- the Citrix client name
- incompatibility warnings

To view these details, enable Extended logging in the runtime settings (F4 Shortcut key) **Log** tab. You can view this information in the **Output** pane or in the **output.txt** file in the script's folder.

Save a Snapshot Bitmap

During recording, the bitmaps generated for the **ctx_sync_on_bitmap** function are saved under the script's **data** folder. The bitmap name has the format of **hash_value.bmp**. If synchronization fails during replay, the generated bitmap is written to the script's output folder, or if you are running it in a scenario, to wherever the output files are written. You can examine the new bitmap to determine why synchronization failed.

Show Vusers

To show Vusers during a scenario, enter the following in the Vuser command line box: **-lr_citrix_vuser_view**. In the Controller, open the Vuser Details dialog box and click **More** to expand the dialog box. Note that this will affect the scalability of the test, so this should only be done to examine a problematic Vuser's behavior.

To reduce the effect on the script's scalability, you can show the details for an individual Vuser by adding the Vuser's ID at the end of the command line: **-lr_citrix_vuser_view <VuserID>**.

To open multiple Vusers, place a comma-separated list of IDs after the command line. Do not use spaces, but you may use commas or dashes. For example, 1,3-5,7 would show Vusers 1,3,4,5, and 7, but would not show Vuser 2, 6, or any Vuser with an ID higher than 7.

When recording with XenApp using a Citrix and Web multi-protocol script, manual modifications may be required to ensure proper recording.

Citrix - Troubleshooting and Limitations

General Limitations

- The Citrix registry patch is installed when you record or replay a Citrix Vuser script for the first time. In rare situations, the error log will indicate that it could not be installed. In this case, try to install the registry patch manually. The patch can be found at: **<installation_folder>\dat\Enable_Citrix_API.reg**. To install the registry patch, double-click **Enable_Citrix_API.reg** on the relevant machine.

Note: The Citrix client is 32-bit software. Therefore, on a 64-bit OS, install this registry patch

under HKLM\SOFTWARE\Wow6432Node\ , and not under HKLM\SOFTWARE\ . To do this, launch Enable_Citrix_API.reg from the 32-bit file manager or modify it before launching it from Windows Explorer."

- Running Citrix Vusers on virtual machines may adversely affect performance due to the sharing of physical resources.
- Text recognition may not work correctly for overlapped windows on Windows 2012 servers.
- On Windows 8.1, in replay, the Start menu may not appear after clicking on the Start button.
Workaround: Add another **ctrx_mouse_click** function into the script below the recorded **ctrx_mouse_click** or **ctrx_obj_mouse_click** functions.
- The Citrix agent does not provide support for Java applications on x86 operating systems.
- If an ICA script succeeds in VuGen, but fails when running on a Load Generator, check the display on the Load Generator machine. If it displays a Citrix dialog box entitled, ICA Client File Security, then, in the Access section of the dialog box, select **Full Access** and **Never ask me again for any application**. Click **OK** to apply your changes.

Note: You can also set these options in **WebICA.ini**. For details, see:
<http://support.citrix.com/article/CTX568194>

- The Citrix agent cannot capture text from Java-based applications or from Internet Explorer 9 and later.
- The Citrix agent cannot capture text from certain Java controls with overlapping text boxes, such as dropdown and combo boxes.
- The recording window size option does not work properly with the Plugin for Hosted Applications 11. The size of the client window is captured, but the server screen resolution is not. This is a Citrix Client limitation and may be fixed in future Citrix Client versions.

Workaround: When recording, set the window size equal to the local screen resolution. When replaying/load testing, set the VuGen or Load Generator screen resolution to equal the resolution used when the script was recorded. To verify the recorded resolution, view the Window property in the **<Script Folder>\default.cfg** file.

- The Citrix Connection Center may prevent record and replay of Citrix ICA scripts, if it is running in a different user session on the same machine.

Workaround: Close all instances of the **concenter.exe** process for all users. To prevent the Citrix Connection Center from starting automatically, set the *ConnectionCenter* registry key to an empty value. This key can be found at:

For 32-bit systems: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

For 64-bit systems: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\Curr

- On a machine with Citrix Receiver 14.4 installed, it is not possible to connect to the Citrix server during record/replay for a single-protocol Citrix script, if connection details have been specified manually in **Recording Options > Citrix > Login**. (This may also apply for later versions of Citrix Receiver.)

There is no issue with Citrix Receiver 14.4 for Citrix single-protocol scripts with ICA files, or Citrix multi-protocol scripts.

- On a machine with Citrix Receiver 13.0-14.6 installed, **ctrx_mouse_move()** does not work.

Workaround: Do one of the following:

- Change the script to use an alternative UI flow. That is, replace mouse-move operations with other operations, such as keyboard operations.
- Upgrade the Citrix Receiver client (both locally and on the Load Generator machine) to Citrix Receiver version 4.7. Then modify the following Windows registry keys to enable the solution:

OS	Registry keys
32-bit	Key location: LOCAL_MACHINE\SOFTWARE\Citrix\ICA Client\CCM
64-bit	<ul style="list-style-type: none">Key location: LOCAL_MACHINE\SOFTWARE\Wow6432Node\Citrix\ICA Client\CCMName: AllowMouseEventType: REG_DWORDValue: 1

Effects and Memory Requirements of Citrix Agent

When you run Citrix Vusers with the agent installed, each Vuser runs its own process of **ctrxagent.exe**. This results in a slight reduction in the number of Vusers that can run on the server machine (about 7%).

When the agent is installed, the memory requirements per Citrix Vuser is approximately 4.35 MB. To run 25 Vusers, you would need 110 MBs of memory.

Random Failures of Functions Accessing Citrix Agent

Communication between Citrix Server and Citrix client-side software is directed via Citrix ICA Virtual Channels. This is a bi-directional connection for the exchange of packet data.

Each Vuser opens its own instance of HPE Citrix Agent on the server side, and, respectively, its own virtual channel. Citrix Virtual Channels may become unreliable under high load. As a consequence, functions that rely on Citrix Agent API (**ctrx_get_text()**, **ctrx_sync_on_obj_info()** etc.) may fail randomly.

Workaround: Use a TCP channel for communication with the Citrix Agent. Set the following flags:

TCPChannel=1 in the [CITIRX] section of the script's **default.cfg** configuration file,

TcpChannelEnabled=1 in the [ChannelConfig] section of the **CtrxAgent.ini** file.

Note that for **MinPortValue** and **NumPorts** flags in CtrxAgent.ini, the agent tries to find a free port and enumerates NumPorts ports starting from MinPortValue. If you have firewall software on the Citrix server or load generator, make sure to configure it to allow connections on these ports.

Citrix Agent will not start

If the Citrix Agent does not start, check that corresponding keys are present in the registry.

In order to be launched during session initialization, Citrix Agent's installer writes it to registry. For servers, it adds it under HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon, and for client machines under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.

Also, make sure the Citrix agent's installation folder, usually **C:\Program Files (x86)\HPE\Agent for Citrix Server**, is set to "Read and Execute" and not only "Read".

Unexpected Disconnection

If you experience "unexpected disconnect" errors, try the following:

- If you suspect this is due to a network issue, you can try the Citrix Session Reliability feature (this can be enabled by the Citrix administrator on the server side). When Session Reliability is enabled, Citrix Client reconnects to the server when the network connection is restored without need for user re-authentication, i.e., transparently for LoadRunner.
- Sometimes the "unexpected disconnect" error may be caused by discrepancy of the script and server timeout settings. Consider the following scenario: The script executes some synchronization function, for example, **ctrl_sync_on_window()**, and waiting time is quite long, say, 180 seconds. The script does not perform any action like mouseclicks or keypresses while it is waiting for the window to appear, and the server disconnects the session when Idle Session Timeout (2 minutes by default) is exceeded. As a result, an "unexpected disconnect" message appears in the replay log. If you get "unexpected disconnect" at the synchronization step, it is recommended to check waiting time value in the script, and session timeouts on the server.

Another workaround for unexpected disconnect at the synchronization step, is to enable User Activity Simulation - **Runtime settings > Citrix > Synchronization > Enable user activity simulation**. If the feature is turned on, LoadRunner will simulate user activity on a Citrix server over the specified time period and in this way prevent a disconnect.

- It may be a result of connecting to a session that already exists on the server. When a Vuser enters an existing session, it cannot receive Windows events from a Citrix ICA object. This is a limitation of the Citrix software. To prevent this, ask the Citrix administrator to configure sessions on the Citrix server to be terminated immediately after disconnect or log off. In the VuGen script side, make sure to add a **ctrl_logoff()** function at the end of the script (in the vuser_end section).

To minimize the risk of entering an existing session, Citrix Agent tries to close the session on the server when communication with the client machine is lost. This functionality is available in Citrix Agent version 12.51 and later, and enabled by default. To disable it, set **LogoffSessionOnExit=0** in CtrxAgent.ini.

Citrix Receiver—Security Warning

The Citrix client may prompt you with a warning "An online application is attempting to access files in your computer". This dialog box blocks the replay because it requires user intervention.

Workaround: To prevent this, configure the registry on the Citrix client machine to allow it to silently access local drives, as described in <http://support.citrix.com/article/CTX124921>.

Failed to get session from client

This error occurs when the Citrix registry patch (LR\dat\Enable_Citrix_API.reg) is not installed

Workaround: Make sure the AllowSimulationAPI key is present in the above registry and not set to 0, as it enables Citrix ICO functionality. Note that in 64-bit operating systems, these keys should reside under the **HKLM\Software\Wow6432Node**, node, since the Citrix client is a 32-bit application.

Citrix Error 13 "Unsupported Function"

The Citrix Error 13 is a general error code that usually refers to an error for which Citrix do not provide a specific code. This error is most common in Performance Center and BPM environments where Citrix processes (wfica32.exe, wfcrun32.exe, concentr.exe, receiver.exe...) are running in sessions other than that of the **mdrv** process.

Workaround: Use TaskManager or ProcessExplorer to find and kill all of these processes.

Citrix Error 70, Client Error 1030 "Protocol driver error"

This error may occur for several reasons: network issues, proxy configuration, and so forth. Often, it occurs when you are running a Citrix+Web multi-protocol script recorded against a secured (https) Web Interface site, and the certificate required by this site is missing on the Load Generator machine.

Workaround: Try to open the published application from the Web Interface on the problematic machine. Look at the log file %APPDATA%\ICAClient\wfcwin32.log and search for "SSL Error 61". If you find this text, it is clearly a certificate issue. For example,

```
09-18-2014 10:28:55:380 Calculator MUCFARMEXT01: SSL Error 61: You have not chosen to trust "AddTrust External CA Root", the issuer of the server's security certificate.
```

Compare certificates on the Load Generator and VuGen machines and install the missing one. Make sure that the attributes match—do not rely on a matching certificate name only. You must also check also other attributes such as "Expiration date".

Capturing Empty Text

In certain Windows 7 installations, VuGen is unable to capture the actual text during recording. Instead it captures empty text.

Workaround:

1. Open **Start > Control Panel > System and Security > System > Advanced system settings**. The System Properties dialog box opens.
2. Select the **Advanced** tab and click the **Settings** button in the **Performance** section.
3. In the Performance Options dialog box, click the **Visual Effects** tab.
4. Clear the check box adjacent to the last option, **Use visual styles on windows and buttons**.

See also:

- For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

Click & Script Protocols

Note: From LoadRunner version 12.00 and later, Web (Click & Script) is only supported for replay—not recording.

Click & Script Protocols - Overview

The Ajax (Click & Script) protocol records Web sessions on a user-action GUI level. VuGen creates a GUI-level script that intuitively represents actions in the Web interface. For example, VuGen generates a **web_button** function when you click a button to submit information, and VuGen generates a **web_edit_field** function when you enter text into an edit box.

Click & Script Vusers support non-HTML code such as Javascript on the client side. VuGen creates an intuitive script that emulates your actions on the web page, and executes the necessary Javascript code.

Click & Script Vusers handle most correlations automatically, reducing the time required to create the script. In most cases, you do not need to define rules for correlations or perform manual correlations after the recording.

Click & Script Vusers allow you to generate detailed Business Process Reports which summarize the script. For example, when you click a button to submit data, VuGen generates a **web_button** function. If the button is an image, VuGen generates a **web_image_submit** function. In the following example, a Vuser clicks the **Login** button.

```
web_image_submit("Login",  
    "Snapshot=t4.inf",  
    DESCRIPTION,  
    "Alt=Login",  
    "Name=login",  
    "FrameName=navbar",  
    ACTION,  
    "ClickCoordinates=31,6",  
    LAST);}
```

The next section illustrates a user navigating to the Asset ExpressAdd process under the Manage Assets branch. The user navigates by clicking the text links of the desired branches, generating **web_text_link** functions.

```
web_text_link("Manage Assets_2",  
    DESCRIPTION,
```

```
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);
web_text_link("Use",
    DESCRIPTION,
    "Text=Use",
    "FrameName=main",
    ACTION,
    "UserAction=Click",
    LAST);
web_text_link("Asset ExpressAdd",
    DESCRIPTION,
    "Text=Asset ExpressAdd",
    "FrameName=main",
    ACTION,
    "UserAction=Click",
    LAST);
```

In the following example, the **web_list** function emulates the selection of a list item.

```
web_list("Year",
    DESCRIPTION,
    "Name=Year",
    "FrameName=CalFrame",
    ACTION,
    "Select=2000",
    LAST);
```

When you click on an image that is associated with an image map, VuGen generates a **web_map_area** function.

```
web_map_area("map2_2",
    DESCRIPTION,
    "MapName=map2",
    "Ordinal=20",
    "FrameName=CalFrame",
    ACTION,
    "UserAction=Click",
    LAST);
```

Note: Click & Script Vusers do not support applets or VB script. If the Web site that is accessed by the Vusers contains these items, use the Web - HTTP/HTML protocol.

Click & Script Recording Tips

This section lists tips for recording click-and-script Vuser scripts.

Note: Some of the items below apply to specific click-and-script protocols only.

Enable the Functional Testing Agent add-on

If you want to record or edit a click-and-script Vuser script in Internet Explorer, you need to enable the HPE Functional Testing Agent add-on in the browser.

If the browser does not prompt you to enable the add-on, enable it through the Manage Add-ons dialog box (**Tools > Manage add-ons**) in Internet Explorer.

Use the Mouse and not the Keyboard

It is preferable to click on an object with the mouse rather than using the keyboard. During recording, use only GUI objects that are within the browser's pane. Do not use any browser icons, controls, the Stop button, or menu items, such as **View > Refresh**. You may, however, use the Refresh, Home, Back and Forward buttons and the address bar.

Do not Record Over an Existing Script

It is best to record into a newly created script—not an existing one.

Avoid Context Menus

Avoid using context menus during recording. Context menus are right-click menus which pop up when clicking certain objects in a graphical user interface.

Avoid Working in Another Browser While Recording

While recording, do not work in any browser window other than the browser windows opened by VuGen.

Wait for Downloads

Wait for all downloads to complete before doing any action, such as clicking on a button or filling in a text field.

Wait for Pages to Load

During recording, it is best to wait for the page to load completely before doing the next step. If you did not wait for all of the pages to load, record the script again.

Navigate to the Start Page

If the last page in an action does not contain the links and buttons that were available at the start of the iteration, then the next iteration will fail. For example, if the first page has a text link **Book A Flight**, make sure to navigate to the appropriate page at the end of your recording, so that the same link will be visible at the end of the business process.

Use a Higher Event Configuration Level

Record the business process again using the **High** event configuration level. For more information on changing the event configuration level, see ["Click & Script Troubleshooting and Limitations" on page 466](#).

Disable Socket Level Recording

In certain cases, the capturing of the socket level messages disrupts the application. For most recordings, socket level data is not required. To prevent the recording of socket level data, disable the option in the recording options. For more information, see ["GUI Properties > Advanced Recording Options" on page 174](#).

Enable the "Record rendering-related property values" Option

If the client-side scripts of the application use a lot of styling activities, enable the **Record rendering-related property values** option before recording the script. For example, enable this option to record additional DOM properties such as **offsetTop**. Note that enabling this option may decrease the recording speed. You can enable the option by selecting **Recording Options > GUI Properties > Advanced**. For more information, see ["GUI Properties > Advanced Recording Options" on page 174](#).

Click & Script - Replay Tips

This section lists tips for replaying click-and-script Vuser scripts.

Note: Some of the items below apply to specific click-and-script protocols only.

Do not re-order statements in a recorded script

Do not change the order of the statements within a recorded script. Also, copying segments of code from one Action to another is not recommended.

Convert non-ASCII characters

If your links contain non-ASCII characters, you should instruct VuGen to convert the data to or from the UTF-8 format.

Enable UTF-8 conversion

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate the **Convert from/to UTF-8** option and set it to **Yes**.

Alternatively, view the list of options that is displayed when a link is not found. Enter the displayed text as-is, such as the hex escape sequences \xA0 or any other non-standard format.

Run the same sequence of actions twice

In some cases, you can perform a certain process only once—such as deleting a user from the database.

Replay will fail after the first iteration because the action is no longer valid. Verify that your business process can be repeated more than once with the same data.

Set unique image properties

In the Step Navigator, double click on the previous image step to open its properties. If the **Id**, **Name**, and **Alt** properties are empty, provide further identification of the image, such as its file name in the **Src** property.

Alternatively, you can add an **Ordinal** argument to specify the occurrence number of the image on that page. The **Ordinal** argument uniquely identifies each image on the page where all other identification arguments are not unique. For more information, see the Function Reference (**Help > Function Reference**).

Check the step's description

If you receive a **GUI Object is not found** error, check the Output pane for a list of the objects in the problematic step. In some cases, the object description changes slightly from run to run.

There are several solutions:

- If the new value is stable, open the script in the Editor and manually modify the value of the step's DESCRIPTION argument.
- If the description changes from run to run, you can use a regular expression in the DESCRIPTION argument. For more information, see the Function Reference (**Help > Function Reference**).
- Alternatively, replace the problematic object description property, such as Name, with the Ordinal property. For more information, see the Function Reference (**Help > Function Reference**).

ThreadingModel

Replay of COM script in VuGen fails when the dll registration is missing the **ThreadingModel** string under the **InprocServer32** folder of the GUID.

Click & Script Miscellaneous Tips

The following additional tips may help you in troubleshooting problems that you experience with click-and-script Vuser scripts.

Note: Some of the items below apply to specific click-and-script protocols only.

Search for Warnings

Search for warnings or alerts in the Output pane.

Verify the Response

Verify the response of the previous step is correct using **web_reg_find**. For more information, see the Function Reference (**Help > Function Reference**).

Use Alternate Navigation

For problematic steps or those using Java applets, use **Alternative Navigation** to replace the Web step with an HTTP level step. Note that the HTTP level steps may require manual correlations. To perform Alternative Navigation, select a step in the **Step Navigator**, or the text in Script View, and select **Replace with alternative navigation** from the right-click menu.

Working with the Kerberos Protocol

If you are using the Kerberos Protocol for authentication, you must customize VuGen to properly convene authorization sessions. Advanced users can attempt to perform this customization themselves.

In order for the Kerberos Protocol to work properly, create a krb5.ini file and put it in an available folder. Save the full path name of krb5.ini into the KRB5_CONFIG environment variable.

The krb5.ini file should contain detailed information about each domain (KDS and AS addresses) and trust chains.

For more information, contact HPE software support.

Click & Script Enhancements

The following section describes several enhancements that can assist you in creating your script.

Most of the features described below are enhancements to the API functions. For detailed information about the functions and their arguments, see the [Function Reference](#) (**Help > Function Reference**) or click F1 on any function.

Adding conditional steps

The Web (Click & Script) functions, **web_xxxx**, allow you to specify conditional actions during replay. Conditions are useful, for example, if you need to check for an element and perform an action only if the element is found.

For example, suppose you perform an Internet search and you want to navigate to all of the result pages by clicking Next. Since you do not know how many result pages there will be, you need to check if there is a Next button, indicating another page, without failing the step. The following code adds a verification step with a notification—if it finds the Next button, it clicks on it.

```
While (web_text_link("Next",  
DESCRIPTION,  
    "Text=Next",  
    VERIFICATION,  
    "NotFound=Notify",  
    ACTION, "UserAction=Click",  
    LAST) == LR_PASS);
```

For details about the syntax and use of the VERIFICATION section, see the [Function Reference](#) (**Help > Function Reference**).

Checking a page title

In **web_browser** steps, you can use the title verification recording option to make sure that the correct page is downloaded. You can instruct the Vuser to perform this check automatically for every step or every navigation to a new top level window.

In addition, you can manually add title verifications to your script at the desired locations, using both exact and regular expression matches.

```
web_browser("test_step",  
DESCRIPTION,  
...  
VERIFICATION,  
    "BrowserTitle=Title",  
    ACTION,]  
,  
LAST);
```

For more information, see the [Function Reference \(Help > Function Reference\)](#).

You can set title verification options directly from within the Recording options. For more information, see the section about recording with Click & Script.

Text check verification

Using text checkpoints, you can verify that a text string is displayed in the appropriate place on a Web page or application and then perform an action based on the findings. You can check that a text string exists (**ContainsText**), or that it does not exist (**DoesNotContainText**), using exact or regular expression matching.

For example, suppose a Web page displays the sentence "Flight departing from New York to San Francisco". You can create a text checkpoint that checks that the words "New York" are displayed between "Flight departing from" and "to San Francisco". (In this example, you would need to use regular expression criteria.)

To implement these checkpoints, you add the Text Check related arguments to the VERIFICATION section of the step. During replay, Vusers search the innerText of the browser's HTML document and any child frames. The **NotFound** argument specifies the action to take if verification fails, either because the object was not found or because the text verification failed: Error, Warning, or Notify.

You can manually add text verifications to your script for existing steps. Place the text verification after the step that generated the element.

The text validation arguments are valid for the following Action functions: **web_browser**, **web_element**, **web_list**, **web_text_link**, **web_table**, and **web_text_area**.

Note: You can only use the same type of text verification once per step (for example, **ContainsText** twice). If you want to check for multiple texts, separate them into several steps. You can, however, use different verifications in the same step (for example, **ContainsText** =;

DoesNotContainText). In this case, all conditions have to be met in order for the step to pass.

In the following example, the verification arguments check that we were not directed from www.acme.com to the French version of the website, acme.com/fr.

```
web_browser("www.acme.com",  
    ACTION,  
    "Navigate=http://www.acme.com/",  
    LAST);  
web_browser("Verify",  
    VERIFICATION,  
    "ContainsText=Go to Acme France",  
    "DoesNotContainText=acme.com in English",  
    LAST);
```

Saving a Java script value to a parameter

The **EvalJavaScript** argument lets you evaluate Java Script on the Web page.

Suppose you want to click on a link which has the same name as the page title. The following example evaluates the document title and uses it in the next `web_text_link` function.

```
web_browser("GetTitle",  
    ACTION,  
    "EvalJavaScript=document.title;",  
    "EvalJavaScriptResultParam=title",  
    LAST);  
web_text_link("Link",  
    DESCRIPTION,  
    "Text={title}",  
    LAST);
```

Working with custom descriptions

Suppose you want to randomly click a link that belongs to some group. For example, on **hpe.com** you want to randomly select a country. Regular description matching will not allow this type of operation. However, using a custom description argument, you can identify the group with an attribute that is common to all the links in the group.

Using the custom description argument, you specify any attribute of the element, even those that are not predefined for that element. During replay, the Vuser searches for those attributes specified in the DESCRIPTION section. Replay will not fail on any unknown argument in the DESCRIPTION section.

For example, to find the following hyperlink:

```
<a href="yahoo.com" my_attribute="bar">Yahoo</a>
```

use:

```
web_text_link("yahoo",  
    DESCRIPTION,  
    "Text=yahoo",  
    "my_attribute=bar",  
    LAST);
```

In the following example, since all the relevant links have the same class name, newmerc-left-ct, you can perform a random click using the following code:

```
web_text_link("Click",  
    DESCRIPTION,  
    "Class=newmerc-left-ct",  
    "Ordinal=random",  
    LAST);
```

The following functions do not support the custom description arguments: **web_browser**, **web_map_area**, **web_radio_group**, and **web_reg_dialog**.

Copying text to the clipboard

VuGen lets you copy text from a browser to the clipboard. This functionality is available in both the Page view and Page Source view. For details on how to copy the text to the clipboard, see ["Work with Snapshots" on page 279](#).

Click & Script API Notes

This section lists general notes about the Web functions.

Regular expressions

You can specify a regular expression for most object descriptions, by preceding the text with "/RE" before the equals sign. For example:

```
web_text_link("Manage Assets",  
    DESCRIPTION,  
    "Text/RE=(Manage Assets)|(Configure Assets)",  
    ACTION,  
    "UserAction=Click",  
    LAST);
```

See the Function Reference (**Help > Function Reference**) for more details.

Ordinals

The Ordinal attribute is a one-based index to distinguish between multiple occurrences of objects with identical descriptions. In the following example, the two recorded **web_text_link** functions have identical arguments, except for the ordinal. The ordinal value of 2, indicates the second occurrence.

```
web_text_link("Manage Assets",
```

```
DESCRIPTION,  
  "Text=Manage Assets",  
  "FrameName=main",  
  ACTION,  
  "UserAction=Click",  
  LAST);  
web_text_link("Manage Assets_2",  
  DESCRIPTION,  
  "Text=Manage Assets",  
  "Ordinal=2",  
  "FrameName=main",  
  ACTION,  
  "UserAction=Click",  
  LAST);
```

Empty strings

There is a difference between not specifying an argument and specifying it as an empty string. When you do not specify an argument, VuGen uses the default value or ignores it. When you list an argument, but assign it an empty string as a value, VuGen attempts to find a match with an empty string or no string at all. For example, omitting the id argument instructs VuGen to ignore the id property of the HTML element. Specifying "ID=" searches for HTML elements with no id property or with an empty ID.

```
web_text_link("Manage Assets_2",  
  DESCRIPTION,  
  "Text=Manage Assets",  
  "Id=",  
  "FrameName=main",  
  ACTION,  
  "UserAction=Click",  
  LAST);
```

Click & Script Troubleshooting and Limitations

This section describes troubleshooting and limitations for click-and-script protocols.

Note: Some of the items below apply to specific click-and-script protocols only.

Recording Issues and Limitations

Browser support

- Only Internet Explorer is supported for Click & Script protocols. To record browser activity on Firefox, use the Web (HTTP/HTML) protocol.

Language Support

- Recording an application in a specific language (e.g., French, Japanese) must be performed on a machine whose default locale (in **Settings > Control Panel > Regional Options**) is the same language
- Support of right-to-left languages is limited (e.g., bi-directional or reversed text may not be processed as expected). This is defined by the default operating system translation table.
- The locale of the load-generator machine, must be configured to be the same as that of the recording machine. It cannot be assumed that the Linux default character set is the same as in Windows, even for US-English machines, and this has to be explicitly verified. For example, the default character set on Linux, is UTF-8.

Application behaves differently while being recorded

If your application behaves differently during recording, than it does without recording, you should determine if the recording problem is unique to Web. The effect may be that a Web page will not load, part of the content is missing, a popup window does not open, and so forth.

Create a new Web (HTTP/HTML) script and repeat the recording.

In the event that the recording fails in Web (HTTP/HTML), we recommend that you disable socket level recording (see ["Click & Script Recording Tips" on page 459](#)).

The problem may be the result of an event listener. Use trial and error to disable event listeners in the **Web Event Configuration** Recording Options, and then re-record your session as a Web user.

Certain Click & Script steps do not generate properly

After recording a script, if not all steps are correctly generated, the problem may be due to the **Windows Component > Internet Explorer Enhanced Security Configuration**.

Remove **Internet Explorer Enhanced Security Configuration** by selecting **Control Panel > Add or Remove Programs > Add or Remove Windows Components** and re-record your script.

Disable an Event Listener

1. Click **Record > Recording Options** to open the Recording Options dialog box.
2. Select the **GUI Properties > Web Event Configuration** node.
3. Click **Custom Settings** and expand the **Web Objects** node. Select an object.
4. Select **Disabled** from the list in the **Record** column for the relevant Web object. If the recording still does not work, enable the listener you previously disabled, and try disabling another one. Repeat these steps until your recording succeeds.

Dynamic menu navigation was not recorded

A dynamic menu is a menu that dynamically changes depending on where you select it. If the dynamic menu navigation was not recorded, record again using "high" event configuration mode. These settings can be found in the **Recording Options > GUI Properties > Web Event Configuration** node.

Certain user actions were not recorded

Check if there is a Java applet running inside the browser. If not, record the script with the Web (HTTP/HTML) protocol.

Replay Issues

GUI object not found

Does the error occur at the beginning of the second iteration?

If the error occurs at the beginning of the second iteration's Action section, it is probably the result of a starting page that was present for the first iteration, but missing for the second one. If the last page in an action does not contain the links and buttons that were available at the start of the iteration, then the next iteration will fail. For example, if the first page has a text link **Book A Flight**, make sure to navigate to the appropriate page, so that the same link will be visible at the end of the business process.

Is it a text link containing non-ASCII characters?

If the problem occurs with non-ASCII characters, you should instruct VuGen to convert the data to a suitable character set.

Enable Data Conversion on Windows Machines

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate **Charset Conversions by HTTP** in the Web (Click & Script) > General options, and set it to **Yes**.

Enable UTF-8 conversion for Linux Machines

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the **Advanced Options** dialog box.
3. Locate **Convert from/to UTF-8** in the General options and set it to **Yes**

Alternatively, view the list of alternatives that are displayed when a link is not found. Enter the displayed text as-is, such as hex escape sequences \xA0 or any other non-standard format.

Can you run the same sequence of actions twice in the application?

In some cases, you can only perform a certain process once, such as deleting a user from the database. Replay will fail after the first iteration, because the action is no longer valid. Verify that your business process can be repeated in the application more than once with the same data, without recording again.

Were the image properties 'Id', 'Name' and 'Alt' empty?

In the **Step Navigator**, double click on the previous image step to open its properties. If the **Id**, **Name**, and **Alt** properties are empty, provide further identification of the image, such as its file name in the **Src** property.

Alternatively, you add an **Ordinal** argument to specify the occurrence number of the image on that page. The **Ordinal** argument uniquely identifies each image on the page where all other identification arguments are not unique. For more information, see the Function Reference (**Help > Function Reference**).

Did the step's description change?

Check the Output pane for a list of the objects in the problematic step. In some cases, the object description changes slightly from run to run.

There are several solutions:

- If the new value is stable, open the Script View and manually modify the value of the step's DESCRIPTION argument(s).
- If the description changes from run to run, you can use a regular expression in the DESCRIPTION argument(s). For more information, see the [Function Reference \(Help > Function Reference\)](#).
- Alternatively, replace the problematic object description property, such as Name, with the Ordinal property. For more information, see the [Function Reference \(Help > Function Reference\)](#).

Did the page load completely during recording?

During recording, it is best to wait for the page to load completely before doing the next step. If you did not wait for all of the pages to load, record the script again.

Replay failure

If the replay is failing at a particular step, check the step description. VuGen sometimes reads a single space as a double space. Make sure that there are no incorrect double spaces in the string.

Miscellaneous Issues

Out of memory error in JavaScript

Increase the JavaScript memory in the Runtime settings.

Increase the JavaScript Memory Size

1. Select **Replay > Runtime Settings** and select the **Internet Protocol > Preferences** node.
2. Click **Options** to open the Advanced Options dialog box.
3. Locate the **Memory Management JavaScript Runtime Memory Size (Kb)** and **Memory Management JavaScript Stack Memory Size (Kb)** options.
4. Increase the memory sizes to 512Kb or higher.

VuGen displays JavaScript errors

If VuGen displays JavaScript errors in the Output pane, enable IE (Internet Explorer) script errors in order to verify that the Javascript itself does not contain errors.

Show Script Errors

1. Open Internet Explorer.
2. Select **Tools > Internet Options** and click the **Advanced** tab.
3. Under **Browsing**, select the **Display a notification about every script error** check box.
4. Rerun the application in IE. If IE displays script errors, then there is a problem with the JavaScript application. If it is not possible to fix the application, you can safely ignore the corresponding replay errors.

Problems following parameterization

If you encounter problems only after you have parameterized values, verify that the values are valid for your application. Perform business process with the value of the parameter and verify that the application accepts it.

Problems with applications that utilize styling actions

If the client-side scripts of the application use a lot of styling activities, you should record the script again after enabling the **Record rendering-related property values** option. This enables the recording of additional DOM objects.

Enable the "Record rendering-related property values" Option

1. Select **Recording > Recording Options** and select the **GUI Properties > Advanced** node.
2. Select the **Record rendering-related property values** check box.

Re-record the Vuser script.

COM/DCOM Protocol

COM/DCOM Protocol Overview

When you record COM client applications, VuGen generates functions that describe COM client-server activity. The recorded script contains interface declarations, API calls and instance calls to methods. Each COM function begins with an **lrc** prefix. You can configure the programming language in which to create a Vuser script as either C or Visual Basic.

For each COM/DCOM Vuser script, VuGen creates the following:

- Interface pointer and other variable declarations in the interfaces.h file.
- Function calls that you can record in the vuser_init, actions or vuser_end sections of the Vuser file.
- A user.h file containing the translation of the Vuser script into low level calls.

COM/DCOM Technology Overview

This section provides an outline of COM technology. This should be enough to get you started with COM Vuser scripts. See Microsoft Developer's Network (MSDN) and other documentation for further details.

COM (Component Object Model) is a technology for developing reusable software components ("plugins"). DCOM (Distributed COM) allows use of COM components on remote computers. Microsoft transaction servers (MTS), Visual Basic and Explorer all use COM/DCOM technology. Thus, the application you are testing may use COM technology indirectly, even though you don't know it. You will probably have to include some, but certainly not all, of the COM calls made by your application in the Vuser script.

Objects, Interfaces, and Type Libraries

COM objects are binary code modules. Each COM object implements one or more interfaces that allow client programs to communicate with it. You need to know about these interfaces in order to follow the COM calls in the Vuser scripts. Type libraries, used as a reference for accessing COM interface methods and parameters, contain descriptions of COM objects and interfaces. Each COM class, interface, and type library is identified by a Global Unique Identifier (GUID).

COM Interfaces

A COM interface provides a grouped collection of related methods. For example, a **Clock** object may have **Clock**, **Alarm** and **Timer** interfaces. Each interface has one or more methods. For example the **Alarm** interface may have **AlarmOn** and **AlarmOff** methods.

An interface may also have one or more properties. Sometimes, the same function may be performed by calling a method or by setting or getting the value of a property. For example, you can set the **Alarm Status** property to **On** or call the **AlarmOn** method.

A COM object may support many interfaces. The **IUnknown** interface is implemented by all components and is used to find out about other interfaces. Many components also implement the **IDispatch** interface, which exposes all other interfaces and methods of the object, allowing implementation of COM automation in scripting languages.

COM Class Context and Location Transparency

COM objects can run on the same machine as the client application, or on a remote server. COM objects that an application creates may be in a local library, a local process or a remote machine ("Remote Object Proxy"). The location of the COM object, known as the "Context," can be transparent to the application. Most users apply the Vusers to check the load on remote servers. Therefore, objects accessed by Remote Object Proxy are usually the most relevant for these tests.

COM Data Types

COM also provides several special data types, including safe arrays, BSTR strings and variants. You may need to use these data types for debugging, parameterization and similar tasks.

COM/DCOM Vuser Script Structure

VuGen COM Vuser scripts are structured in a special way to meet the needs of COM interfaces.

Interface Methods

Calls to interface methods have the following names and syntax conventions:

```
lrc_<interface name>_<method name>(instance,...);
```

Note that the instance is always the first parameter passed.

The vendors of the respective COM components usually supply documentation for the interface functions.

Interface Pointers

The interface header file defines the interface pointers, as well as other variables, that can be used in the script. Each interface has an Interface ID (IID) which uniquely identifies the interface.

The format of the interface definition is:

```
<interface type>*<interface name> = 0; ///{<IID of the interface type>}"
```

In the following example, the interface type is IDispatch, the name of the interface instance is IDispatch_0, and the IID of IDispatch type is the long number string:

```
IDispatch* IDispatch_0= 0;///"{00020400-0000-C000-000000000046}"
```

Vuser Script Statements

The COM Vuser script consist of code that creates object instances, retrieves interface pointers and calls the interface methods. Each user action may generate one or more COM calls. Each COM call is coded by VuGen as a group of statements. Each such group is contained in a separate scope enclosed in braces. Several different statements prepare for the main call by assigning values and performing type conversions. For example, the group of calls needed to create an object may look like this:

```
{
GUID pClsid = lrc_GUID("student..1");
IUnknown * pUnkOuter = (IUnknown*)NULL;
unsigned long dwClsContext = lrc_ulong("7");
GUID riid = IID_IUnknown;
lrc_CoCreateInstance(=;pClsid, pUnkOuter, dwClsContext, =;riid, (void**)=>IUnknown_0,
CHECK_HRES);
}
```

Error Checking

Each COM method or API call returns an error value. VuGen will set a flag to check or not to check errors during replay, depending upon whether the call succeeded during the original recording. The flag appears as the last argument of the function call and has these values:

CHECK_HRES	This value is inserted if the function passed during recording and errors should be checked during replay.
DONT_CHECK_HRES	This value is inserted if the function failed during recording and errors should not be checked during replay.

COM Sample Vuser Scripts

This section shows examples of how VuGen emulates a COM client application. It is divided up into the basic COM script operations. Each type of operation is done within a separate scope.

Instantiation of the Object

To use a COM object, the application must first instantiate it and get a pointer to an interface of that object.

VuGen does the following to instantiate an object

1. VuGen calls `lrc_GUID` to get a unique ProgID for the object, to be stored in `pClsid`:

```
GUID pClsid = lrc_GUID("student..1");
```

pClsid is the unique global CLSID of the object, which was converted from the **student.student.1** ProgID.

2. If the unknown interface pointer is a pointer to an aggregated object, VuGen retrieves the pointer to that object, or else it sets it to NULL:

```
IUnknown * pUnkOuter = (IUnknown*)NULL;
```

3. VuGen sets the contexts of the object to be created:

```
unsigned long dwClsContext = lrc_ulong("7");
```

dwClsContext contains the context of the object (in process, local, remote or combinations of these.)

4. VuGen sets a variable to hold the requested interface ID, which is `IUnknown` in this case:

```
GUID riid = IID_IUnknown;
```

riid contains the interface ID of the **IUnknown** interface.

5. After the input parameters are prepared, a call to `lrc_CoCreateInstance` creates an object using the parameters defined in the preceding statements. A pointer to the `IUnknown` interface is assigned to output parameter `IUnknown_0`. This pointer is needed for subsequent calls:

```
lrc_CoCreateInstance(=;pClsid, pUnkOuter, dwClsContext, =;riid, (void**)
=;IUnknown_0, CHECK_HRES);
```

The input parameters were prepared and explained above. Since the call succeeded, VuGen sets error checking on during the user simulation by inserting the **CHECK_HRES** value. The call returns a pointer to the `IUnknown` interface in **IUnknown_0**, that can be used in subsequent calls.

Retrieving an Interface

After creating an object, VuGen has access only to the **IUnknown** interface. VuGen will use the **IUnknown** interface for communicating with the object. This is done using the **QueryInterface** method of the **IUnknown** standard interface. The first parameter in a VuGen method call is the interface instance. In this case it is the **IUnknown_0** pointer set previously by **CoCreateInstance**. The **QueryInterface** call requires as input the ID of the interface to be retrieved, and returns a pointer to the interface designated by that ID.

Get the Interface

1. First, VuGen sets a parameter, riid, equal to the ID of the Istudent interface:

```
GUID riid = IID_Istudent;
```

A call to QueryInterface assigns a pointer to the Istudent interface to output parameter Istudent_0 if the Istudent object has such an interface:

- 2.

```
Irc_IUnknown_QueryInterface(IUnknown_0, =;riid, (void**)=;Istudent_0, CHECK_HRES);
```

Using an Interface to Set Data

Here is an example of using the methods of the interface to set data. Suppose that in the application, the user is supposed to input a name. This activates a method for setting the name. VuGen records this in two statements. One statement is used for setting up the name string and the second one sets the name property.

Set up the Entire Function Call

1. First, VuGen sets a variable (Prop Value) equal to the string. The parameter is of type BSTR, a string type used in COM files:

```
BSTR PropValue = lrc_BSTR("John Smith");
```

In subsequent stages, you will probably parameterize this call, replacing "John Smith" with a parameter, so that different names are used each time the Vuser script is run.

2. Next, VuGen calls the Put_Name method of the Istudent interface to enter the name:

```
Irc_Istudent_put_name(Istudent_0, PropValue, CHECK_HRES);
```

Using an Interface to Return Data

Returning data from an application is different than entering the data, because you might want to store these values and use them as inputs in subsequent calls for parameterization.

The following is an example of what VuGen may do when the application retrieves data

1. Create a variable of the appropriate type (in this case a BSTR) that will contain the value of the property.

```
BSTR pVal;
```

2. Get the value of the property, in this case a name, into the **pVal** variable created above, using the `get_name` method of the **Istudent** interface in this example.

```
lrc_Istudent_get_name(Istudent_0, =>pVal, CHECK_HRES);
```

3. VuGen then generates a statement for saving the values.

```
//lrc_save_BSTR("param-name",pVal);
```

The statement is commented out. You can remove the comments and change <param-name> to a variable with a meaningful name to be used for storing this value. VuGen will use the variable to save the value of **pVal** returned by the previous call. You can then use the variable as a parameterized input in subsequent calls to other methods.

The IDispatch Interface

Most COM objects have specific interfaces. Many of them also implement a general-purpose interface called **IDispatch**, which VuGen translates in a special way. **IDispatch** is a "superinterface" that exposes all of the other interfaces and methods of a COM object. Calls to the **IDispatch:Invoke** method from VuGen scripts are implemented using **lrc_Disp** functions. These calls are constructed somewhat differently from calls to other interfaces.

The **IDispatch** interface **Invoke** method can execute a method, it can get a property value, or it can set a value or reference value for a property. In the standard **IDispatch:Invoke** method these different uses are signaled in a **wflags** parameter. In the VuGen implementation they are implemented in different procedure calls that invoke a method or put or get a property.

For example, a call to **IDispatch** to activate the `GetAgentsArray` method may look like this:

```
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray", /*locale*/1033, LAST_ARG, CHECK_HRES);
```

The parameters in the above call are:

IDispatch_0	This is the pointer to the IDispatch interface returned by a previous call to the IUnknown:Queryinterface method.
--------------------	---

GetAgentsArray	This is the name of the method to invoke. Behind the scenes, VuGen will get the ID of the method from the name.
1033	This is the language locale.
LAST_ARG	This is a flag to tell the IDispatch interface that there are no more arguments.
CHECK_HRES	This flag turns on checking of HRES, since the call succeeded when it was recorded.

In addition, there might be another parameter, **OPTIONAL_ARGS**. This signals that in addition to any standard parameters, VuGen is sending some optional arguments. Each optional argument consists of a pair giving the ID or name of the argument and its value. For example, the following call to **Irc_DispatchMethod** passes optional arguments "#3" and "var3":

```
{
    GUID riid = IID_IDispatch;
    Irc_IOptional_QueryInterface(IOptional_0, =;riid, (void**)=;IOptional_0,
CHECK_HRES);
}
{
    VARIANT P1 = lrc_variant_short("47");
    VARIANT P2 = lrc_variant_short("37");
    VARIANT P3 = lrc_variant_date("3/19/1901");
    VARIANT var3 = lrc_variant_scode("4");
    Irc_DispatchMethod((IDispatch*)IOptional_0, "in_out_optional_args",
/*locale*/1024, =;P1, =;P2, OPTIONAL_ARGS, "#3", =;P3, "var3", =;var3, LAST_ARG,
CHECK_HRES);
}
```

The different **Irc_Dispatch** methods that use the **IDispatch** interface are detailed in the Function Reference (**Help > Function Reference**).

Type Conversions and Data Extraction

As shown in the above example, many COM parameters are defined as variants. To extract these values, VuGen uses a number of conversion functions, derived from the equivalent COM functions. The full list is given in the Function Reference (**Help > Function Reference**). Previously, we showed how the **Irc_DispatchMethod1** call was used to retrieve an array of name strings:

```
VARIANT retValue = lrc_variant_empty();
retValue = lrc_DispatchMethod1((IDispatch*)IDispatch_0, "GetAgentsArray", /*locale*/1033,
LAST_ARG, CHECK_HRES);
```

The following example now shows how VuGen gets the strings out of **retValue**, which is a variant that will be read as an array of strings.

First, VuGen extracts the BSTR array from the variant:

```
BstrArray array0 = 0;  
array0 = lrc_GetBstrArrayFromVariant(=;retValue);
```

With all the values in array0, VuGen provides you with code that you can use to extract the elements from the array for later use in parameterization, as in the example below:

```
//GetElementFrom1DBstrArray(array0, 0); // value: Alex  
//GetElementFrom1DBstrArray(array0, 1); // value: Amanda  
....
```

VuGen has numerous type conversion functions and functions for extracting conventional types from variants. These are detailed in the [Function Reference \(Help > Function Reference\)](#).

Selecting COM Objects to Record

The application you are testing may use a great many COM objects. Only a few may actually create load and may be important for the load test. Thus, before you record a COM application, you should select the objects you want to record for the load test. VuGen allows you to browse for objects from type libraries that it can read on the local machine and on other computers in the network.

Deciding Which Objects to Use

There are several ways to decide which COM objects should be included in the test. Try to determine which remote objects are used by the software. If you are unsure which objects to use, try using the default filter. The Environments branch of the filter includes calls to three sets of objects (ADO, RDS and Remote) that are likely to generate load on remote servers.

You can also check the actual calls to refine the filter. After you have recorded the test, you can save the file and look in the **data** folder that VuGen creates for a file named **lrc_debug_list_<nnn>.log**, where **nnn** is the process number. This log file contains a listing of each COM object that was called by the recorded application, regardless of whether or not the recording filter included that object. Only calls that generate load on the server should be included for recording.

For example, the following is a local COM of the Visual Basic library:

```
Class JetES {039EA4C0-E696-11D0-878A-00A0C91EC756}  
was loaded from type library "JET Expression Service Type Library"  
({2358C810-62BA-11D1-B3DB-00600832C573} ver 4.0)
```

It should not be added since it does not generate load on the server.

Likewise, since the OLE DB and Microsoft Windows Common Controls are local objects, the following are examples of classes and libraries that are not going to place any load on the server and should not be recorded:

```
Class DataLinks {2206CDB2-19C1-11D1-89E0-00C04FD7A829}  
was loaded from type library "Microsoft OLE DB Service Component 1.0 Type Library"  
({2206CEB0-19C1-11D1-89E0-00C04FD7A829} ver 1.0)
```

```
Class DataObject {2334D2B2-713E-11CF-8AE5-00AA00C00905}  
was loaded from type library "Microsoft Windows Common Controls 6.0 (SP3)"  
({831FDD16-0C5C-11D2-A9FC-0000F8754DA1} ver 2.0)
```

However, for example, a listing such as the following indicates a class that should be recorded since it does generate load on the server:

```
Class Order {B4CC7A90-1067-11D4-9939-00105ACECF9A}  
was loaded from type library "FRS"  
({B4CC7A8C-1067-11D4-9939-00105ACECF9A} ver 1.0)
```

Calls to classes of the **FRS** library, used for instance in the flight_sample that is installed with VuGen, use server capacity and should be recorded.

If a COM object itself calls other COM objects, all the calls will be listed in the type information log file. For example, every time the application calls an **FRS** class function, the **FRS** library calls the **ActiveX Data Object (ADO)** library. If several functions in such a chain are listed in a filter, VuGen records only the first call that initiates the chain. If you selected both **FRS** and **ADO** calls, only the **FRS** calls will be recorded.

On the other hand, if you select only the **ADO** library in the filter, then calls to the **ADO** library will be recorded. It is often easier to record the call to the first remote object in the chain. In some cases, however, an application may use methods from several different COM objects. If all of them use a single object that puts a load on the server, you could only record the final common object.

Which Objects Can be Selected

VuGen can only record objects if it can read their type libraries. If the type libraries were not installed in the system or VuGen cannot find them, the COM objects will not be listed in the Recording Options dialog box. If they are used by your application, VuGen will not be able to identify these objects and will identify them as **INoTypeInfo** in the files.

Which Interfaces Can be Excluded

For each object, the Recording Options dialog box will show you all interfaces that are listed in the Type Library, and allow you to specify inclusion or exclusion of each one. However, **ADO**, **RDS** and Remote Objects can be included in the filter as a group. The filter will not show the individual objects of those environments or their interfaces. Objects that you included from type libraries may also have interfaces that are not listed in the type library and therefore not shown in the Recording Options dialog. After generating a VuGen script, you can identify these interfaces in the script and get their GUID numbers from the interfaces.h file that VuGen generates. Using this information, you can exclude the interfaces as explained in ["COM/DCOM > Filter Recording Options" on page 146](#).

Database Protocols

Database Protocols Overview

Suppose that you have a database of customer information that is accessed by customer service personnel located throughout the country. You use Database Users to emulate the situation in which the database server services many requests for information. A Database User could:

- Connect to the server
- Submit an SQL query
- Retrieve and process the information
- Disconnect from the server

VuGen supports the following database types: Oracle and ODBC. The resulting script contains LRD functions that describe the database activity.

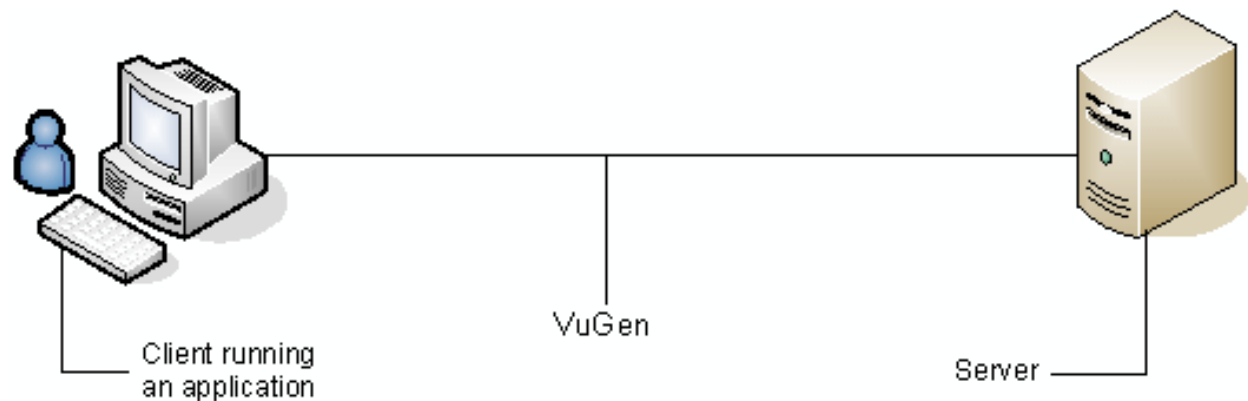
Note: To replay ODBC Protocol scripts on Linux machines, unixODBC v.2.3.1 or higher is required.

See also:

- ["VuGen Database Recording Technology" below](#)
- ["Database Grids" on the next page](#)

VuGen Database Recording Technology

VuGen creates Database Vuser scripts by recording all the activity between a database client and a server. VuGen monitors the client end of the database and traces all the requests sent to and received from the database server.



Like all other Vusers created using VuGen, Database Vusers communicate with the server without relying on client software. Instead, each Database Vuser executes a script that executes calls directly to server API functions.



You create Database Vuser scripts in a Windows environment using VuGen. Once you create a script, you can assign it to Vusers in both Windows and Linux environments.

Users working in a Linux only environment can create Database Vuser scripts through programming using VuGen templates as the basis for a script. For information about programming Database Vuser scripts on Linux, see ["Create and Run Scripts in Linux" on page 848](#).

Database Grids

When you record or replay a Vuser script, the data that is returned by each query is displayed in a data grid. In a Vuser script, the existence of a data grid is indicated by a **GRID** statement. VuGen displays data grids in either the Data Grids pane or the Snapshot pane.

Correlate a value in a data grid

Display the data grid in the Snapshot pane, right-click in a cell inside the data grid, and select **Create Correlation**.

Save the data in a data grid to a file

Display the data grid in the Snapshot pane or the Data Grids pane, right-click in any cell in the data grid, and select **Save Grid To File**.

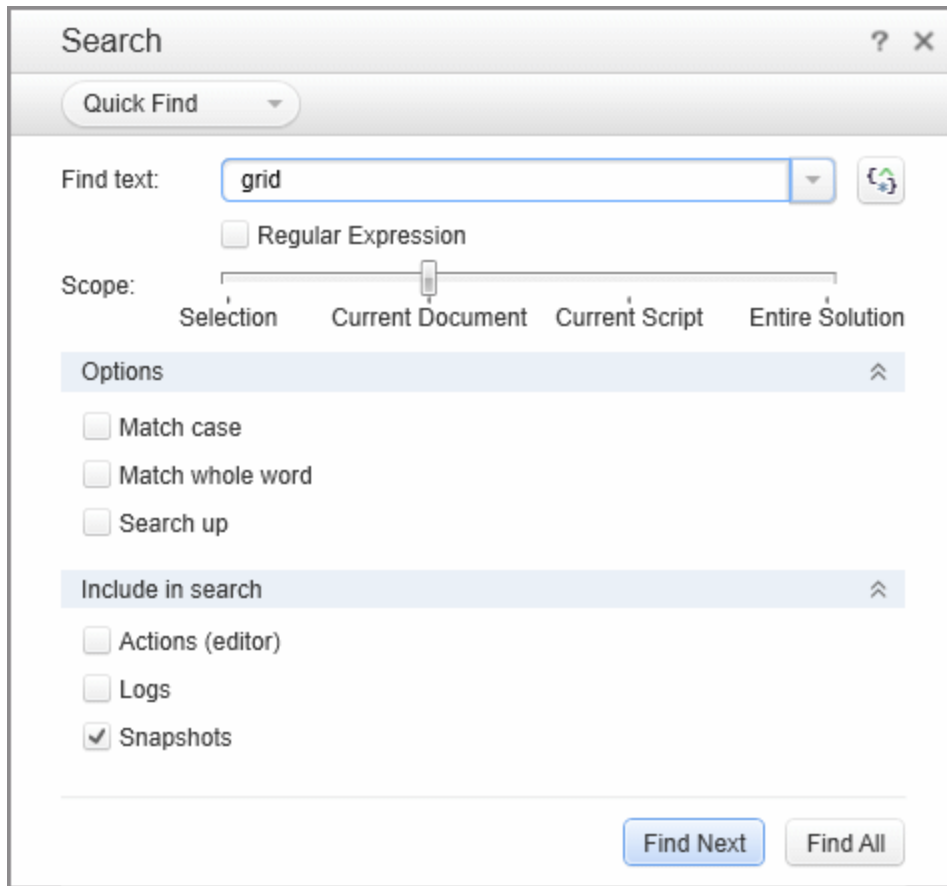
Copy the text from a cell in a data grid to the clipboard

Display the data grid in the Snapshot pane or in the Data Grids pane, right-click in the cell in the data grid, and select **Copy Selection**.

Search for data inside a data grid

1. Display the data grid in the Snapshot pane, and click **Search > Quick Find** to open the Search dialog box.

2. Click **Include in Search**, and then select the **Snapshots** check box.



See also:

- ["Work with Snapshots" on page 279](#)
- ["Snapshot Pane" on page 62](#)

Handling Database Errors

You can control how database Vusers handle errors when you run a database Vuser script. By default, if an error occurs during script execution, the script execution is terminated. To change the default behavior, you can instruct the Vuser to continue when an error occurs. You can apply this behavior in different ways as described below.

Globally Modifying Error Handling

You can change the way that Vusers handle errors by issuing an `LRD_ON_ERROR_CONTINUE` or `LRD_ON_ERROR_EXIT` statement. By default, a Vuser aborts the script execution when it encounters any type of error—database, parameter related, and so on. To change the default behavior, insert the following line into your script:

```
LRD_ON_ERROR_CONTINUE;
```

From this point on, the Vuser continues script execution, even when an error occurs.

You can also specify that the Vuser continue script execution when an error occurs only within a segment of the script. For example, the following code tells the Vuser to continue script execution even if an error occurs in the `lrd_stmt` or `lrd_exec` functions:

```
LRD_ON_ERROR_CONTINUE;  
lrd_stmt(Csr1, "select..."...);  
lrd_exec(...);  
LRD_ON_ERROR_EXIT;
```

Use the `LRD_ON_ERROR_CONTINUE` statement with caution, as significant and severe errors may be missed.

Locally Modifying Error Handling

You can set error handling for a specific function by modifying the severity level. Functions such as **`lrd_stmt`** and **`lrd_exec`**, which perform database operations, use severity levels. The severity level is indicated by the function's final parameter, **`miDBErrorSeverity`**. This parameter tells the Vuser whether or not to continue script execution when a database error occurs (error code 2009). The default, 0, indicates that the Vuser should abort the script when an error occurs.

For example, if you reference a table that does not exist, the following database statement fails and the script execution terminates.

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1 /*Deferred*/,  
1 /*Dflt Ora Ver*/, 0);
```

To tell a Vuser to continue script execution, even when a database operation error occurs for that function, change the statement's severity parameter from 0 to 1.

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1 /*Deferred*/,  
1 /*Dflt Ora Ver*/, 1);
```

When the severity is set to 1 and a database error occurs, a warning is issued. Note that the severity level set for a particular function applies only to that function.

Debugging Database Applications

The following tips apply to database applications only (such as ODBC):

Generating Debugging Information

Note: You can now set options to view most of the information described in this section using VuGen's user interface.

VuGen contains an inspector "engine." You can force VuGen recorder to create "inspector" output by editing `<install_dir>\config\vugen.ini` as follows:

```
[LogMode]
EnableAscii=ASCII_LOG_ON
```

When this option is enabled, VuGen creates a file, `vuser.asc` in the Data folder at the end of the recording. Note that this option should be used for debugging purposes only. This output file can become very large (several MB) and have serious effects on machine performance and disk space.

For cases like ODBC-based applications, it is possible to configure the ODBC Administrator (located in the Windows Control Panel) to provide a similar trace output. Open the ODBC options, and select 'Trace ODBC calls' to ON. Similarly the ODBC Developer Kit provides a Spy utility for call tracing.

To enable further debug information, add the following section to the `<install_dir>\config\vugen.ini`:

```
[INSPECTOR]
TRACE_LEVEL=3
TRACE_FILENAME=c:\tmp\sqltrace.txt
```

The file (`sqltrace.txt`) will include useful internal information regarding the hooking calls made during recording. The `trace_level` is between 1 and 3, with 3 representing the most detailed debug level. Note that in VuGen versions 5.02 and higher, you can set the trace level from the user interface.

Examining Compiler Information

You can view information about each stage of code generation, preprocessing and compilation to determine the source of any errors.

Code Generation Information

Look at the `vuser.log` file under the Data folder. This file, which contains a log of the code generation phase, is automatically created at the end of every lrd recording (i.e. all database protocols).

The following is an example of a log file:

```
lrd_init: OK
lrd_option: OK
lrd_option: OK
lrd_option: OK
Code generation successful
lrd_option: OK
```

1rd_end: OK

If any of the messages are not OK or successful, then a problem occurred during the code generation.

Preprocessing and Compilation Information

During runtime, VuGen displays information about both the preprocessing and compilation processes.

Database Protocols - Troubleshooting and Limitations

This section describes troubleshooting and limitations for database protocols.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

Troubleshooting all database protocols

IE crashes when recording Oracle NCA or Oracle - Web scripts

This can occur due to an incompatible dll file.

Replace the incompatible dll

1. Open the **C:\program files\oracle\Jinitiator_1.3.1.18\bin\hotspot** folder.
2. Back up the **jvm.dll** file.
3. Delete the **jvm.dll** file and replace it with a different version of the file.

Evaluating Error Codes

When a Vuser executes an LRD function, the function generates a return code. A return code of 0 indicates that the function succeeded. For example, a return code of 0 indicates that another row is available from the result set. If an error occurs, the return code indicates the type of error. For example, a return code of 2014 indicates that an error occurred in the initialization.

There are four types of return codes, each represented by a numerical range:

Type of Return Code	Range
Informational	0 to 999
Warning	1000 to 1999
Error	2000 to 2999
Internal Error	5000 to 5999

For more detailed information on the return codes, see the Function Reference (**Help > Function Reference**).

You can evaluate the return code of an LRD function to determine if the function succeeded. The following script segment evaluates the return code of an `lrd_fetch` function:

```
static int rc;  
rc=lrd_fetch(Csr15, -13, 0, 0, PrintRow4, 0);  
if (rc==0)  
    lr_output_message("The function succeeded");  
else  
    lr_output_message("The function returned an error code:%d",rc);
```

Two-tier Database Scripting Tips

The following section offers solutions for two-tier database scripts.

Question 1: Why does the script fail when it is data driven, while the same values work with the application itself?

Answer: The failure may be a result of trailing spaces in your data values. Even though the data values that you type directly into the GUI are probably truncated, you should manually eliminate them from your data file. Tab-delimited files can hide trailing spaces and therefore obscure problems. In general, comma-delimited files are recommended. You can view the files in Excel to see if things are correct.

Question 2: Why does an SQL error of an invalid cursor state occur on the second iteration?

Answer: The **`lrd_close_cursor`** function may not have been generated or it may be in the *end* section instead of the *action* section. You will need to add a cursor close function or move it from the *end* section to make the script iterate successfully.

Opening a new cursor may be costly in terms of resources. Therefore, we recommend that you only open a cursor once in the *actions* section during the first iteration. You can then add a new parameter that contains the iteration number as a string by using the Iteration Number type. Call this parameter *IterationNum*. Then, inside the *actions* section replace a call to open a new cursor, for example,

```
lrd_open_cursor(=;Csr1, Con1, 0);
```

with

```
if (!strcmp(lr_eval_string("<IterationNum>"), "1"))  
    lrd_open_cursor(=;Csr1, Con1, 0);
```

Question 3: How can I fix code produced by VuGen that will not compile because of data declarations in the *vdf.h* file?

Answer: The problem, most likely, is an SQL data type that is not supported by VuGen. For Microsoft SQL, you can often work around this issue by replacing the undefined error message in *vdf.h* with "DT_SZ" (null terminated string). Although this is not the actual datatype, VuGen can compile the script correctly. Please report the problem and send the original script to customer support.

Question 4: What is the meaning of LRD Error 2048?

Answer: VuGen is failing because it is trying to bind a variable with a longer length than what was allocated during recording. You can correct this by enlarging the variable definition in *vdf.h* to receive a longer string back from the database. Search this file for the unique numeric identifier. You will see its definition and length. The length is the third element in the structure. Increase this length as required and the script will replay successfully.

For example, in the following script, we have:

```
lrd_assign(=;_2_D354, "<ROW_ID>", 0, 0, 0);
```

In *vdf.h*, we search for *_2_D354* and find

```
static LRD_VAR_DESC _2_D354 = {  
    LRD_VAR_DESC_EYECAT, 1, 10, LRD_BYTYPE_ODBC,  
    {0, 0, 0}, DT_SZ, 0, 0, 15, 12};
```

We change it to:

```
static LRD_VAR_DESC _2_D354 = {  
    LRD_VAR_DESC_EYECAT, 1, 12, LRD_BYTYPE_ODBC,  
    {0,, 0}, DT_SZ, 0, 0, 15, 12};
```

The complete definition of *LRD_VAR_DESC* appears in *lrd.h*. You can find it by searching for `typedef struct LRD_VAR_DESC`.

Question 5: How can I obtain the number of rows affected by an UPDATE, INSERT or DELETE when using ODBC and Oracle?

Answer: You can use **lrd** functions to obtain this information. For ODBC, use **lrd_row_count**. The syntax is:

```
int rowcount;  
  
.  
.  
.  
  
lrd_row_count(Csr33, =;rowcount, 0);
```

Note that **lrd_row_count** must immediately follow the pertinent statement execution.

For Oracle you can use the fourth argument of **lrd_exec**.

```
lrd_exec(Csr19, 1, 0, =;rowcount, 0, 0);
```

If you are using Oracle's OCI 8, you can use the fifth argument of **lrd_ora8_exec**.

```
lrd_ora8_exec(OraSvc1, OraStm3, 1, 0, =;uliRowsProcessed, 0, 0, 0, 0, 0);
```

Question 6: How can I avoid duplicate key violations?

Answer: Occasionally, you will see a duplicate key violation when performing an Insert. You should be able to find the primary key by comparing two recordings to determine the problem. Check whether this or earlier UPDATE or INSERT statement should use correlated queries. You can use the data dictionary in order to find the columns that are used in the violated unique constraint.

In Oracle you will see the following message when a unique constraint is violated:

```
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

In this example SCOTT is the owner of the related unique index, and PK_EMP is the name of this index. Use SQL*Plus to query the data dictionary to find the columns. The pattern for this query is:

```
select column_name from all_ind_columns where index_name = '<IndexName>' and index_
owner = '<IndexOwner>';
select column_name from all_ind_columns where index_name = 'PK_EMP' and index_owner =
'SCOTT';
```

Since the values inserted into the database are new, they might not appear in earlier queries, but they could be related to the results of earlier queries, such as one more than the value returned in an earlier query.

You can use the Query Analyzer to find out which columns used by the key or index. The pattern for this query is:

```
select C.name
  from sysindexes A, sysindexkeys B, syscolumns C
 where C.colid = B.colid and C.id = B.id and
       A.id = B.id and A.indid = B.indid
       and A.name = '<IndexName>' and A.id = object_id('<TableName>')
select C.name
  from sysindexes A, sysindexkeys B, syscolumns C
 where C.colid = B.colid and C.id = B.id and
       A.id = B.id and A.indid = B.indid
       and A.name = 'IX_newtab' and A.id = object_id('newtab')
```

If you still encounter problems, be sure to check the number of rows changed for Updates and Inserts for both recording and replay. Very often, an UPDATE fails to change any rows during replay, because the WHERE clause was not satisfied. This does not directly result in an error, but it causes a table not to be properly updated, and can cause a later SELECT to select the wrong value when correlating the query.

Also verify that there are no problems during multi-user replay. In certain instances, only one user will successfully perform an UPDATE. This occurs with Siebel, where it is necessary to manually write a loop to overcome the problem.

Question 7: The database does not appear to be modified after replaying a script which should have modified the database.

Answer: Through the user application's UI, check if the updated values appear when trying to see the current data accessible to the application. If the values have not been updated, you need to determine they were not changed. Possibly, an UPDATE statement changed one or more rows when the application was recorded, and did not change any during replay.

Check these items:

- **Verify statement.** If there is a WHERE clause in the UPDATE statement, verify that it is correct.
- **Check for correlations.** Record the application twice and compare the UPDATE statements from each of the recordings to make sure that the necessary correlations were performed.
- **Check the total number of rows.** Check the number of rows that were changed after the UPDATE. For Oracle, this information is stored in the fourth parameter of **lrd_exec**. For ODBC, use **lrd_row_count** to determine the number of rows updated. You can also add code to your script that prints the number of rows that were updated. If this value is 0, the UPDATE failed to modify the database.
- **Check the SET clause.** Check the SET clause of the UPDATE statement. Make sure that you correlated any necessary values here instead of hard-coding them. You can see this by comparing two recordings of the UPDATE.

In certain cases, the UPDATE works when replaying one Vuser, but not for multiple Vusers. The UPDATE of one Vuser might interfere with that of another. Parameterize each Vuser so that each one uses different values during the UPDATE, unless you want each Vuser to update with the same values. In this case try adding retry logic to perform the UPDATE a second time.

Question 8: How do I avoid the unique column name error when replaying a statement recorded with an Oracle Application. For example:

```
lrd_stmt(Csr9, "SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "  
"MTL_UNITS_OF_MEASURE "  
"WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "  
"SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

The following error message was issued:

```
"lndo.c/fjParse: "oparse" ERROR return-code=960, oerhms=ORA-00960:  
ambiguous column naming in select list".
```

Answer: Change the statement by adding an alias to at least one of the non-unique columns, thereby mapping it to a new unique name. For example:

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION FROM"  
"MTL_UNITS_OF_MEASURE "  
"WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "  
"SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Troubleshooting Oracle 2-Tier Vusers

This section contains a list of common problems that you may encounter while working with Oracle Vusers, and suggested solutions.

ORA-20001 and ORA-06512

Errors ORA-20001 and ORA-06512 appear during replay when the lrd_stmt contains the pl/sql block: fnd_signon.audit_responsibility(...)

This statement fails during replay because the sign-on number is unique for each new connection.

Solution

In order to solve this problem you need to use the new correlation tool for the sign-on number. This is second assigned value in the statement.

After you scan for possible values to correlate, highlight the value of the second lrd_assign_bind() for the failed statement. Note that the values in the "correlated query" window may not appear in the same order as the actual recorded statements.

The grid containing the substitution value should appear after the lrd_stmt which contains the pl/sql block: fnd_signon.audit_user(...).

Note: Since the sign-on number is unique for every connection, you need to use correlation for each new connection that you record.

Example of Solution

The following statement failed in replay because the second value, "1498224" is the unique sign-on number for every new connection.

```
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility  
(:s,:l,:f,:a,:r,:t,:p)"  
"; end;", -1, 1, 1, 0);  
lrd_assign_bind(Csr6, "s", "D", =;s_D216, 0, 0, 0);  
lrd_assign_bind(Csr6, "l", "1498224", =;l_D217, 0, 0, 0);  
lrd_assign_bind(Csr6, "f", "1", =;f_D218, 0, 0, 0);
```

```
lrd_assign_bind(Csr6, "a", "810", =;a_D219, 0, 0, 0);  
lrd_assign_bind(Csr6, "r", "20675", =;r_D220, 0, 0, 0);  
lrd_assign_bind(Csr6, "t", "Windows PC", =;t_D221, 0, 0, 0);  
lrd_assign_bind(Csr6, "p", "", =;p_D222, 0, 0, 0);  
lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

The sign-on number can be found in the lrd_stmt with "fnd_signon.audit_user". The value of the first placeholder "a" should be saved. The input of "a" is always "0" but the output is the requested value.

Modified code:

```
lrd_stmt(Csr4, "begin fnd_signon.audit_user(:a,:l,:u,:t,:n,:p,:s); end;",  
-1, 1, 1, 0);  
lrd_assign_bind(Csr4, "a", "0", =;a_D46, 0, 0, 0);  
lrd_assign_bind(Csr4, "l", "D", =;l_D47, 0, 0, 0);  
lrd_assign_bind(Csr4, "u", "1001", =;u_D48, 0, 0, 0);  
lrd_assign_bind(Csr4, "t", "Windows PC", =;t_D49, 0, 0, 0);  
lrd_assign_bind(Csr4, "n", "OraUser", =;n_D50, 0, 0, 0);  
lrd_assign_bind(Csr4, "p", "", =;p_D51, 0, 0, 0);  
lrd_assign_bind(Csr4, "s", "14157", =;s_D52, 0, 0, 0);  
lrd_exec(Csr4, 1, 0, 0, 0, 0);  
lrd_save_value(=;a_D46, 0, 0, " saved_a_D46");  
Grid0(17);  
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility  
(:s,:l,:f,:a,:r,:t,:p)"  
"; end;", -1, 1, 1, 0);  
lrd_assign_bind(Csr6, "s", "D", =;s_D216, 0, 0, 0);  
lrd_assign_bind(Csr6, "l", " <saved_a_D46>", =;l_D217, 0, 0, 0);  
lrd_assign_bind(Csr6, "f", "1", =;f_D218, 0, 0, 0);  
lrd_assign_bind(Csr6, "a", "810", =;a_D219, 0, 0, 0);  
lrd_assign_bind(Csr6, "r", "20675", =;r_D220, 0, 0, 0);  
lrd_assign_bind(Csr6, "t", "Windows PC", =;t_D221, 0, 0, 0);
```

```
lrd_assign_bind(Csr6, "p", "", =;p_D222, 0, 0, 0);  
lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

Working with large numbers

Large numbers (NUMBER data type) sometimes appear in different format in the GRID and in the ASCII file. This difference makes it more difficult to identify numbers while searching for values to save for correlation.

For example, you could have a value appear as 1000003 in the grid, but as 1e+0006 in the Recording Log (ASCII file).

Workaround

If you have an error during replay and the correlation tool cannot locate the value in previous results, look for this value in the other format in grid.

ORA-00960

This error may occur with non-unique column names. For example:

```
lrd_stmt(Csr9, "SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "  
"MTL_UNITS_OF_MEASURE "  
"WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "  
"SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

In this case you receive the following error:

```
"lrd.c/fjParse: "oparse" ERROR return-code=960, oerhms=ORA-00960:  
ambiguous column naming in select list".
```

Workaround

Change the statement by adding an alias to at least one of the non-unique columns, thus mapping it to a new unique name. For example:

```
lrd_stmt(Csr9, "SELECT UOM_CODE, UOM_CODE second, DESCRIPTION FROM"  
"MTL_UNITS_OF_MEASURE "  
"WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "  
"SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Alternate Workaround: remove ORDER BY from the lrd statement.

ORA-2002

Error 2002 appears when you try to use an unopened cursor. It occurs when you replay a user more than one iteration and you recorded into more than one section of the script.

Specifically, if a cursor is opened in the *vuser_init* section and closed in the Actions section, then you will encounter this error on the second iteration if you try to use the cursor. This is because it was closed but not re-opened.

For example: You have *lrd_open_cursor* in the *vuser_init* section and *lrd_close_cursor* in the Actions section. If you replay this user more than one iteration, you are going to get an error in the second iteration because you try using an unopened cursor (it was closed in the first iteration, but not re-opened in the second).

Workaround

The easiest way to solve this is to move the **lrd_close_cursor** or/and **lrd_close_connection** of the problem cursor to the *vuser_end* section.

Database Protocols (lrd)

Replay of recorded asynchronous operations is not supported.

Wrong Client Version

You may receive an error message when running the wrong Oracle client version:

```
"Error: lrdo_open_connection: "olog" LDA/CDA return-code_019: unable to allocate memory in the user side"
```

Workaround

You need to modify the library information in the *lrd.ini* file, located in the product's *bin* folder. This file contains the settings that indicate which version of database support is loaded during recording or replay. The file contains a section for each type of host.

For example, the following section of the *lrd.ini* file is for Oracle on Windows NT:

[ORACLE_WINNT]

```
805=lrdo32.dll+ora805.dll
816=lrdo32.dll+oci.dll
815=lrdo32.dll+oraclient8.dll
804=lrdo32.dll+ora804.dll
803=lrdo32.dll+ora803.dll
73=lrdo32.dll+ora73.dll
72=lrdo32.dll+ora72.dll
71=lrdo32.dll+orant71.dll
```

These settings indicate that Vusers should use the **ora805.dll** library if the client uses Oracle 8.0.5, oci.dll for Oracle 8.1.6, and so on.

Flex (RTMP/AMF) Protocol

Flex Overview

Note: This topic applies to Flex Vuser scripts only.

The Flex Vuser protocol emulates communication between a client server application that uses the Flex collection of technologies.

What is Flex?

Flex is a collection of technologies that provides developers with a framework for building RIAs (Rich Internet Applications) based on the Flash Player.

RIAs are lightweight online programs that provide users with more dynamic control than with a standard Web page. Like Web applications built with Ajax, Flex applications are generally more responsive, because the application does not need to load a new Web page every time the user performs an action. However, unlike working with Ajax, Flex is independent of browser implementations such as JavaScript or CSS. The framework runs on Adobe's cross-platform Flash Player.

Flex applications consist of many MXML and ActionScript files. They are compiled into a single SWF movie file which can be played by the Flash player installed on the client's browser.

Note: For Flex applications working with SOAP data, use the **Web Services** Vuser protocol.

Flex Technologies

The following tables describe the specific technologies that are supported by the VuGen recording engine.

Technology	Description
AMF0	Action Message Format
AMF3	Action Message Format - Compressed format
RTMP	Real Time Message Protocol: Messaging and streaming over TCP
RTMPS	Real Time Message Protocol: Messaging and streaming over TCP/SSL
RTMPT	Real Time Message Protocol Tunneled: Messaging and streaming over HTTP

VuGen supports the following development solutions:

Development Platforms	Description
BlazeDS	Open Source Remoting and Messaging solution
GraniteDS	Open source development and integration solution for building Flex applications.
LiveCycle	Adobe development and integration solution for building Flex applications.

Learn more about developing a Flex Vuser script

Topic	Description
Creating a Flex Vuser script	See VuGen's generic documentation about creating Vuser scripts ["Creating Vuser Scripts - Overview" on page 113].
Recording	<p>In addition to the generic documentation about recording Vuser scripts ["Recording - Overview" on page 131], see:</p> <ul style="list-style-type: none"> • "Record a Flex Script" on page 506 • "Setting the Flex Recording Mode" on page 507 • "RTMP/RTMPT Streaming" on page 498 • "RTMP Tunneled" on page 505 <p>You may need to configure recording options for your Flex script:</p> <ul style="list-style-type: none"> • "Flex > RTMP Recording Options" on page 162 • "Flex > Externalizable Objects Recording Options" on page 163
Correlating	<p>In addition to the generic VuGen documentation on correlating Vuser scripts ["Correlation Overview" on page 232], see:</p> <p>"Flex Correlations" on page 511</p>
Replaying	<p>In addition to the generic VuGen documentation about replaying Vuser scripts ["Replay Overview" on page 277], see:</p> <ul style="list-style-type: none"> • "Query an XML Tree" on page 513 <p>You may need to configure runtime settings for your Flex script:</p> <ul style="list-style-type: none"> • Flex > RTMP view • Flex > Externalizable view • Flex > Configuration view

Topic	Description
Debugging	See the generic documentation about debugging Vuser scripts ["Debugging Overview" on page 313]. <ul style="list-style-type: none"> • "Externalizable Objects in Flex Scripts" on page 509 • "Serialize Flex Scripts" on page 512
Viewing Test Results	See the generic documentation about viewing test results ["Replay Summary Pane" on page 108].

Recording Flex Scripts

When you record a Flex application, VuGen generates Flex Vuser script functions that emulate the application. The following tables describe the functions that are supported by the Flex protocol.

AMF

VuGen's **Flex** protocol lets you create scripts that emulate Flex applications working with AMF0 and AMF3.

Function Name	Description
flex_amf_call	Sends an AMF request.
flex_amf_define_envelope_header_set	Defines a set of envelope headers.
flex_amf_define_header_set	Defines a set of AMF headers.
flex_login	Logs on to a password-protected Flex application.
flex_logout	Logs off of a password-protected Flex application.
flex_ping	Checks if a Flex application is available.
flex_remoting_call	Invokes one or more methods of a server-side Remote object (RPC).

AMF Example 1:

In the following example, **flex_ping** checks for the availability of a service. The **flex_remoting_call** function invokes the service remotely.



Example:

```
flex_ping("1",
    "URL=http://<HOST>/weborb.aspx",
```



```
"Snapshot=t6.inf",
LAST);
flex_remoting_call("getProductEdition::GenericDestination",
"URL=http://testlab1/weborb30/console/weborb.aspx",
"Snapshot=t1.inf",
INVOCATION,
"Target=/2",
"Operation=getProductEdition",
"Destination=GenericDestination",
"DSEndpoint=my-amf",
"Source=Weborb.Management.LicenseService",
"Argument=<arguments/>",
LAST);
```

AMF Example 2:

In the following AMF0 example, the **flex_amf_call** function accesses a gateway and sends message to the server.



Example:

```
flex_amf_call("EchoAny",
"Gateway=http://<host>/gateway.aspx",
"Snapshot=t1.inf",
"IsParseResponse=No",
MESSAGE,
"Method=EchoAMF.EchoAMF.EchoAny",
"TargetObjectId=/1",
BEGIN_ARGUMENTS,
"<boolean>true</boolean>",
END_ARGUMENTS,
LAST);
```

AMF Example 3:

In the following AMF3 example, the **flex_remoting_call** function sends the server an AMF call that can be serialized.



Example:

```
flex_remoting_call(
"product::getProductsByName",
URL=http://<HOST>:<PORT>/amf;jsessionid={CorrelationParameter}",
"Snapshot=t1.inf",
"IsParseResponse=No",
INVOCATION,
"Target=/2",
"Operation=getProductsByName",
```



```
"Destination=product",
"DSEndpoint=my-amf",
"DSId=8E3759E5-E51A-3906-0EAB-6119CD1E26BF",
"Argument="
    "<arguments>"
        "<string>A</string>"
    "</arguments>",
LAST);
```

RTMP Functions

Function Name	Description
flex_rtmp_connect	Connects a client to an RTMP server and sets connection options.
flex_rtmp_disconnect	Disconnects a client from an RTMP server.
flex_rtmp_receive_stream	Receives streaming data from an RTMP server.
flex_rtmp_receive	Receives responses from an RTMP server.
flex_rtmp_send	Sends a request to an RTMP server.

RTMP Example

In the following example, **flex_rtmp_receive** receives data.

```
flex_rtmp_receive("recv_step0",
"ConnectionID=19",
"Snapshot=tRTMP44.inf",
CHANNEL,
"ChunkStreamID=2",
CHANNEL,
"ChunkStreamID=2",
LAST);
```

RTMP Tunneled Functions

Function Name	Description
flex_rtmp_tunneled_connect	Connects a client to an RTMP server over HTTP.
flex_rtmp_tunneled_disconnect	Disconnects a client from session over HTTP with an RTMP server.
flex_rtmp_tunneled_send	Sends a request to an RTMP server over HTTP.

RTMP Tunneled Example

In the following example, **flex_rtmp_tunneled_send** sends an RTMP tunneled request.

```
flex_rtmp_tunneled_send(  
    "send_step0",  
    "SessionID=0",  
    "Snapshot=t30.inf",  
    MESSAGE,  
    "DataType=command message amf3",  
    "ChunkStreamID=3",  
    "MessageStreamID=0",  
    "Argument="  
        "<arguments>"  
        ...  
        "</arguments>",  
    LAST);
```

For detailed syntax information about all of the Flex functions, see the [Function Reference \(Help > Function Reference\)](#).

RTMP/RTMPT Streaming

VuGen's Flex protocol supports record and replay of streaming data for both the RTMP and RTMPT protocols. You can record using either the regular recording mode or the simplified recording mode. The simplified mode enables VuGen to generate a single function in place of the multiple functions that are generated when the regular recording mode is used.

VuGen also supports RTMPS and RTMPTS in which the streaming data is sent over SSL.

When you use the simplified mode to record, the following occurs:

- For an RTMP-based stream: VuGen generates a single **flex_rtmp_receive_stream** step in place of many **flex_rtmp_receive** and **flex_rtmp_send** steps.
- For an RTMPT-based stream: VuGen generates a single modified **flex_rtmp_tunneled_send** step in place of many **flex_rtmp_tunneled_send** steps.

The single generated **flex_rtmp_receive_stream** or **flex_rtmp_tunneled_send** step makes the Vuser script more readable (by eliminating multiple lines of code), and makes the script replay more reliable. It is recommended that you use the simplified mode for recording your Vuser scripts, unless the Vuser activity includes asynchronous behavior, as described below.

The simplified recording mode is the default recording mode for streaming. To activate the simplified mode, in the Recording Options dialog box, click **Flex > RTMP** and select **Generate single step for RTMP/T stream handling**.

The differences between the simplified and regular recording modes are listed below:

	Simplified mode	Regular mode
(Recording option) Generate single step for RTMP/T stream handling check box	Selected	Not selected
Functions generated	RTMP: Generates flex_rtmp_receive_stream functions. RTMPT: Generates flex_rtmp_tunneled_send functions.	Generates flex_rtmp_receive and flex_rtmp_send steps.
Number of functions generated per stream	One	Multiple
Supports asynchronous behavior	No	Yes
Default Mode	Yes	No

Note: The simplified recording mode is supported for Flash Media Server versions 3.5 and 4.

Synchronous Vuser behavior

The **flex_rtmp_receive_stream** and **flex_rtmp_tunneled_send** functions that are generated when the simplified recording mode is selected are synchronous functions. This means that no other Vuser functions can be executed while either of these functions is executing. For example, consider a Vuser script that includes a **flex_rtmp_receive_stream** function that streams a video for 5 minutes. During the 5 minute period during which the video is streaming, the Vuser will not be able to perform any other actions, such as clicking the **Pause** button or skipping to a different location in the video. Clicking a button while a video is streaming is an example of asynchronous behavior.

Although a single generated step makes script replay more reliable, it is not able to replay asynchronous actions (such as pause and seek) that you may have performed while recording the script. The single generated step also does not replay the automatic requests that the client performs when Dynamic Stream is in use. If it is important to replay these asynchronous actions, you must record the Vuser script using the regular recording mode - not the simplified recording mode - and then manually modify the generated script as described below.

Modifying scripts to replay asynchronous user actions

If your Vuser script must be able to replay asynchronous actions that are performed while a streaming action is executed, you must record the Vuser script using the regular recording mode - not the simplified recording mode - and then manually modify the generated script. The modified script will include a combination of single streaming steps and the more verbose steps that are generated with regular recording.

Note: In this section, we will use the term **required user actions** to refer to the actions that must be performed while a video is streamed.

To create a script that can replay asynchronous behavior, first you record the script using the regular recording mode - not the simplified recording mode. Thereafter, identify the **flex_rtmp_send** steps that represent required user actions. Then replace the steps between the required user actions with single streaming functions. See the sections below for details.

Note: The modification procedure differs slightly between RTMP and RTMPT steps.

Modifying recorded Flex RTMP steps

When you use the regular recording mode, VuGen generates **flex_rtmp_receive** and **flex_rtmp_send** steps for all communication with the server. This ensures that user actions such as pause and seek, as well as automatic requests that the client performs when Dynamic Stream is in use, are included in the script. However, this method also captures less-necessary lines of code that are difficult to read and may not be reliable during replay of streaming actions.

Note: To activate the regular recording mode, clear the **Generate single step for RTMP/T stream handling** option in the **Flex > RTMP** pane of the Recording Options dialog box.

Follow the instructions below to remove the unnecessary **flex_rtmp_receive** and **flex_rtmp_send** steps from your script.

1. Search for the **flex_rtmp_send** step that contains the initial play argument. For example:

```
flex_rtmp_send("send_step2",  
"ConnectionID=10",  
"Snapshot=tRTMP6.inf",  
MESSAGE,  
...  
MESSAGE,  
...  
"Argument=<arguments><string>play</string><number>0</number><null/>"  
...  
LAST);
```

2. Delete or comment out the **flex_rtmp_receive** steps that occur during streaming. For example:

```
//This is the start of the stream:
```

```
flex_rtmp_receive("recv_step2",  
    "ConnectionID=10",  
    "Snapshot=tRTMP7.inf",  
    CHANNEL,  
    "ChunkStreamID=2",  
    CHANNEL,  
    "ChunkStreamID=2",  
    CHANNEL,  
    "ChunkStreamID=4",  
    CHANNEL,  
    "ChunkStreamID=2",  
    LAST);  
flex_rtmp_receive("recv_step3",  
    "ConnectionID=10",  
    "Snapshot=tRTMP8.inf",  
    CHANNEL,  
    "ChunkStreamID=5",  
    CHANNEL,  
    ...
```

3. Remove the **flex_rtmp_send** steps that are not related to the required user actions, such as "user control message" types. For example:

```
flex_rtmp_send("send_step3",  
    "ConnectionID=10",  
    "Snapshot=tRTMP9.inf",  
    MESSAGE,  
    "DataType=user control message",  
    "EventType=set buffer length",  
    "MessageStreamID=1",  
    "BufferLength=100",  
    LAST);
```

4. When you find a **flex_rtmp_send** step that represents a required user action, do the following:
 - a. Manually add a **flex_rtmp_receive_stream** step before the send step.
 - Make sure that the **ConnectionID** argument has the same value as the steps you removed above it.
 - The **Snapshot** argument is not relevant for the manually added step.
 - You can use the **ContinueToNextStepAfter = <msec>** argument to control the minimum play duration of the stream to download before continuing to the next step.
 - b. Determine the **flex_rtmp_send** steps that represent the required user actions. These will likely include arguments such as **pauseRaw**, **pause**, **seek** and **play2** (for Dynamic Stream). For example:

```
flex_rtmp_send("send_step5",  
"ConnectionID=10",  
"Snapshot=tRTMP62.inf",  
MESSAGE,  
"DataType=command message amf3",  
"ChunkStreamID=8",  
"MessageStreamID=1",  
"Argument=<arguments><string>pauseRaw</string><number>0</number><null/>"  
"<boolean>true</boolean><number>12000</number></arguments>",  
LAST);
```

- c. Determine whether there are some extra **flex_rtmp_send** steps that you can remove. For example, if you dragged a button to seek in the stream, subtle jerks in the motion may be recorded as separate pause and seek actions. In these cases, may not need all of them. Keep only those that describe the desired operations.
 - d. Identify the **flex_rtmp_receive** step that indicates that the server has received the end of the user action. For example:

```
//This is the confirmation from the server on the "seek" command.  
flex_rtmp_receive("recv_step55",  
"ConnectionID=10",  
"Snapshot=tRTMP68.inf",  
CHANNEL,  
"ChunkStreamID=2",  
CHANNEL,
```

```
"ChunkStreamID=2",  
LAST);
```

5. Repeat steps 2 - 4 for each set of unnecessary receive data and required user actions in your script.

For additional details on **flex_rtmp_receive_stream** including a complete example, see the Function Reference (**Help > Function Reference**).

Modifying recorded Flex RTMPT steps

When you use the regular recording mode, VuGen generates a **flex_rtmp_tunneled_send** step for all communication with the server. This ensures that user actions such as pause and seek, as well as automatic requests that the client performs when Dynamic Stream is in use, are included in the script. However, this method also captures less-necessary lines of code that are difficult to read and may not be reliable during replay of streaming actions.

Note: To activate the regular recording mode, clear the **Generate single step for RTMP/T stream handling** check box in the **Flex > RTMP** pane of the Recording Options dialog box.

Follow the instructions below to remove the unnecessary steps from your script.

1. Search for the **flex_rtmp_tunneled_send** step that contains the initial play argument. For example:

```
flex_rtmp_tunneled_send("send_step2",  
    "SessionID=1",  
    "Snapshot=t36.inf",  
    MESSAGE,  
    ...  
    MESSAGE,  
    ...  
    "Argument=<arguments><string>play</string><number>0</number><null/>"  
    ...  
    LAST);
```

2. Remove **flex_rtmp_tunneled_send** steps that are not related to required user actions, such as "user control message" types. For example:

```
flex_rtmp_tunneled_send("send_step3",  
    "SessionID=10",  
    "Snapshot=t15.inf",
```

```
MESSAGE,  
  "DataType=user control message",  
  "EventType=set buffer length",  
  "MessageStreamID=1",  
  "BufferLength=100",  
  LAST);
```

3. When you find a **flex_rtmp_tunneled_send** step that represents a required user action, do the following:
 - a. Add a **ContinueToNextStepAfter = <msec>** argument to the previous step. The **ContinueToNextStepAfter = <msec>** argument controls the minimum play duration of the stream to download before continuing to the next step. For example:

```
flex_rtmp_tunneled_send("send_step2",  
  "SessionID=1",  
  "Snapshot=t36.inf",  
  
  //Read the stream until at least 15 seconds of media have been downloaded  
  "ContinueToNextStepAfter = 15000",  
  MESSAGE,  
  ...  
  MESSAGE,  
  ...  
  "Argument=<arguments><string>play</string><number>0</number><null/>"  
  ...  
  LAST);
```

- b. Determine the **flex_rtmp_tunneled_send** steps that represent the required user actions. These will typically include arguments such as **pauseRaw**, **pause**, **seek** and **play2** (for Dynamic Stream). For example:

```
flex_rtmp_tunneled_send("send_step5",  
  "SessionID=10",  
  "Snapshot=t16.inf",  
  MESSAGE,
```

```
"DataType=command message amf3",  
"ChunkStreamID=8",  
"MessageStreamID=1",  
"Argument=<arguments><string>pauseRaw</string><number>0</number><null/>"  
"<boolean>true</boolean><number>12000</number></arguments>",  
LAST);
```

- c. Determine whether there are extra **flex_rtmp_tunneled_send** steps that you can remove. For example, if you dragged a button to seek in the stream, subtle jerks in the motion may be recorded as separate pause and seek actions. In these cases, you may not need all of them. Keep only those that describe the desired operations.

4. Repeat steps 2 - 3 for each set of unnecessary send data and required user actions in your script.

For additional details on the **flex_rtmp_tunneled_send** function, including a complete example, see the Function Reference (**Help > Function Reference**).

Live Streaming

VuGen's Flex protocol supports Adobe's Live Streaming. If VuGen detects a live stream while you record a Vuser script, VuGen adds '**ContinueToNextStepAfter**' and '**ContinueMode**' arguments to the generated **flex_rtmp_receive_stream** or **flex_rtmp_tunneled_send** function. These additional arguments enable the live stream to be accurately replayed. For details on these arguments, see the Function Reference (**Help > Function Reference**).

Note: The default value of the generated **ContinueToNextStepAfter** argument is the length of time (in milliseconds) for which the video was streamed while the Vuser script was recorded.

RTMP Tunneled

VuGen supports the recording of RTMP Tunneled steps in Flex application which are split into the following step types:

- **Messaging support.** The Flex protocol supports enhanced record and replay of messaging and has been verified for Adobe LiveCycle Data Services ES2 Version 3.1.
- **Streaming support.** The Flex protocol supports enhanced record and replay of streaming. For details, see ["RTMP/RTMPT Streaming" on page 498](#).

When you record a Flex stream, by default, VuGen generates a single **flex_rtmp_tunneled_send** step in place of many **flex_rtmp_tunneled_send** steps. This step makes your script more readable (eliminating tens or hundreds of lines) and makes the replay more reliable.

Note: The new **flex_rtmp_tunneled_send** step is generated when the **Generate single RTMP/T step** option is selected in the **Flex:RTMP** pane of the **Recording Options** dialog box.

Although this step makes the script more reliable, it does not replay certain actions you may perform while recording your script, such as pause and seek. It also does not replay the automatic requests that the client performs when Dynamic Stream is in use.

If it is important to replay these actions, you can clear the **Generate single RTMP/T step** option in the **Flex> RTMP** pane of the **Recording Options** dialog box, which causes LoadRunner to generate the steps for all of the raw streaming data.

However, to ensure proper replay, you must manually modify the generated script as described in ["RTMP/RTMPT Streaming" on page 498](#).

The above functionality has been verified for Flash Media Server versions 3.5 and 4.

- **Externalizable objects.** VuGen supports externalizable objects over RTMP Tunneled. For details, see ["Externalizable Objects in Flex Scripts" on page 509](#).
- **User Data Points.** VuGen generates a number of new data points that provide more useful information for analysis.
- The Flex RTMP-Tunneled protocol supports manual correlation using **web_reg_save_param_xpath API**.

For additional details on **flex_rtmp_tunneled_send** including a complete example, see the Function Reference (**Help > Function Reference**).

Record a Flex Script

Note: You can generate a Flex Vuser script by analyzing an existing network traffic file (capture file). This method may be useful for creating Vuser scripts that emulate activity on mobile applications. For details, see ["Create a Vuser Script by Analyzing a Captured Traffic File" on page 685](#).

1. Create a new script or open an existing script.

Select **New Script and Solution > Flex protocol**. For details, see ["Creating or Opening Vuser Scripts" on page 113](#).

2. Configure the recording options. The recording options affect the way that a Vuser script is generated after recording or regenerating the script.

- **Recording options > Flex > RTMP > Generate single step for RTMPT/Streaming**

This option, selected by default, enables VuGen to create a single step while recording a stream. However, when you create a single step, certain actions are not replayed, such as pause and seek. If you want to be able to replay these actions, disable the option.

- **Recording Options > Flex > Configuration > Use External JVM**

If you are using an external Java Virtual Machine select this option and configure the path of the JVM in the value field.

- **Recording Options > Flex > Configuration > Use GraniteDS**

Select this option if you are using GraniteDS as a sever side Data Service configuration.

- **Recording Options > Flex > Externalizable Objects**

This option enables you to specify additional .jars that are required to record your script. For details, see ["Externalizable Objects in Flex Scripts" on page 509](#).

- **Recording Options > HTTP Properties > Advanced**


Make sure that the **Save snapshot resources locally** option is enabled, as local snapshots are required for data loading. For details, see ["HTTP Properties > Advanced Recording Options" on page 175](#). For concept and user interface details, see ["Recording Options" on page 141](#).

3. Initialize the recording session.

When creating a new script, this occurs automatically. To manually start recording, click **Start Record** on the VuGen toolbar, complete the Start Recording dialog box, and then click **Start Recording**. VuGen's floating toolbar appears, VuGen opens your application and begins recording your actions. For user interface details, see ["Start Recording Dialog Box" on page 221](#). For details on the script sections into which you can record, see ["Vuser Script Sections" on page 132](#).

4. Perform business processes on your application.

Perform the desired business processes that you wish to record. The floating toolbar allows you to insert transactions, rendezvous points, and comments. You can also use the floating toolbar to specify into which section of the script to record. For user interface details, see ["Floating Recording Toolbar" on page 225](#).

Click **Stop**  on the floating toolbar when you are finished recording.

5. Regenerate the script to determine if all the steps were parsed correctly. For details, see ["Code Generation in the Flex Protocol" on the next page](#).

Setting the Flex Recording Mode

You can instruct VuGen how to generate a script from a Flash Remoting session using the Flex and Web Protocols.

Example

Use Web HTTP technology to generate **web_custom_request** functions with the Flash Remoting information.



Example:

```
web_url("flash",
    "URL=http://<HOST>:<PORT>/flash/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    EXTRARES,
    "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
    "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);
```



```
web_link("Sample JavaBean Movie Source",
    "Text=Sample JavaBean Movie Source",
    "Snapshot=t2.inf",
    EXTRARES,
    "Url=XMLExample.swf", "Referer=", ENDITEM,
    "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);
web_custom_request("gateway",
    "URL=http://<HOST>:<PORT>/flashservices/gateway",
    "Method=POST",
    "Resource=0",
    "RecContentType=application/x-amf",
    "Referer=",
    "Snapshot=t3.inf",
    "Mode=HTML",
    "EncType=application/x-amf",
    "BodyBinary=\\x00\\x00\\x00\\x01\\x00\\x10amf_server_debug\\x01
        \\x00\\x00\\x00`\\x03\\x00\\ncoldfusion\\x01\\x01\\x00
        \\namfheaders\\x01\\x00\\x00\\x03amf\\x01\\x00\\x00
        \\x0Bhttpheaders\\x01\\x00\\x00\\trecordset\\x01\\x01
        \\x00\\x05error\\x01\\x01\\x00\\x05trace\\x01\\x01
        \\x00\\x07m_debug\\x01\\x01\\x00\\x00\\t\\x00\\x01
        \\x00/flashgateway.samples.FlashJavaBean.testDocument
        \\x00\\x02/1\\x00\\x00\\x004\\n\\x00\\x00\\x00\\x01
        \\x0F\\x00\\x00\\x00*
        <TEST message=\"test\\\"><INSIDETEST /></TEST>",
    LAST);
```

Code Generation in the Flex Protocol

Code Generation Notification

If a Flex or Java over HTTP script encounters an error during the code generation phase, VuGen issues a warning. This warning appears in the Errors pane, when the **Warnings** button is selected, and the **Define Available Categories** filter is set to **All** or **Code Generation Notification**. The list of warnings displays details about each error, as well as recommended actions for resolving the problem. Follow the recommended actions and regenerate the script.

If the error is related to externalizable objects in a Flex script, see ["Externalizable Objects in Flex Scripts" on the next page](#).

To manually open the Errors pane at any time, select **View > Errors**.

Parsing Responses in Flex Scripts

When generating a Flex script, VuGen attempts to parse responses for any of the following steps:

- flex_amf_call
- flex_remoting_call

- flex_login
- flex_logout
- wflex_ping

If the parsing fails, the following attribute is dynamically added to the step:

```
IsParseResponse = No
```

This instructs VuGen not to parse the responses for that step during script replay. Every time you regenerate the script, VuGen will attempt to parse again, and will set this parameter to false if it fails. If needed, you can delete this line, or set the value to = 'Yes' to force VuGen to parse responses for that step during replay.

Additionally, you can manually add the attribute and set the value to 'No' in a generated script, even if the parse is successful, as it may enhance replay performance.

Externalizable Objects in Flex Scripts

When recording a Flex application, information is usually passed between the client and server using known serialization methods (AMF). If this is the case, VuGen creates a **flex_amf_call** and both the request and response are parsed.

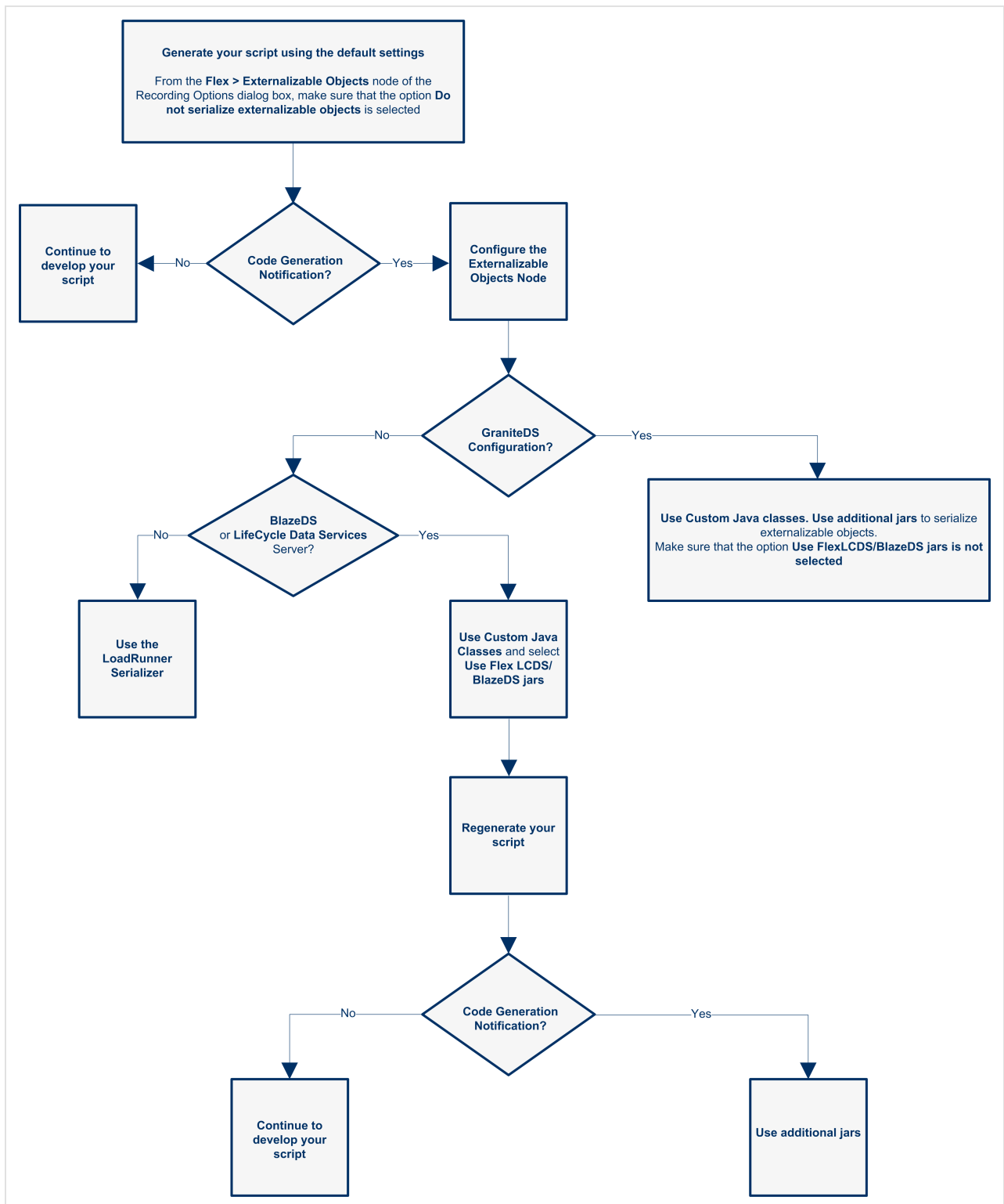
However, when a given AMF object uses a custom serialization method (externalizable), VuGen automatically issues a warning. This warning displays details about the exception, as well as recommended actions for resolving it.

The following are some examples of the exceptions that the generated script may include when an AMF object uses a custom serialization method:

- **Request and response not parsed.** This exception is automatically displayed in the Errors pane when the **Warnings** button is selected, and the **Define Available Categories** filter is set to **All** or **Code Generation Notification**. Details about the exception are listed, as well as recommended actions. For details, see ["Errors Pane" on page 72](#).
- **Request parsed but response is not parsed.** VuGen generates a `IsParseResponse=No` statement. Additionally, VuGen issues a warning that is automatically displayed in the Errors pane when the **Warnings** button is selected, and the **Define Available Categories** filter is set to **All** or **Code Generation Notification**. The list of warnings displays details about the exception, as well as recommended actions. For details, see ["Errors Pane" on page 72](#).

For details on configuring the **Recording Options > Flex > Externalizable Objects** Node, see ["Flex > Externalizable Objects Recording Options" on page 163](#).

The following flowchart illustrates the steps to resolve externalizable objects in Flex scripts:



See also:

- ["Flex > Externalizable Objects Recording Options" on page 163](#)
- ["Serialize Flex Scripts" on the next page](#)

Flex Correlations

VuGen supports correlation in Flex scripts.

Support for correlations applies to the following Flex steps:

- flex_login
- flex_logout
- flex_ping
- flex_amf_call
- flex_remoting_call
- flex_rtmp_tunnelled_connect
- flex_rtmp_tunnelled_send

Flex correlation includes integration with the following features:

- **Correlations rules**
DSid, **jsessionId**, and **RTMPT ID**
- **Design Studio**
- **Manual correlation** using the API **web_reg_save_param_xpath**.

For general information, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

For task details, see ["Correlate Scripts Using Design Studio" on page 250](#).

Flex Snapshots

Vuser scripts based on the Flex protocol utilize VuGen's Snapshot pane.

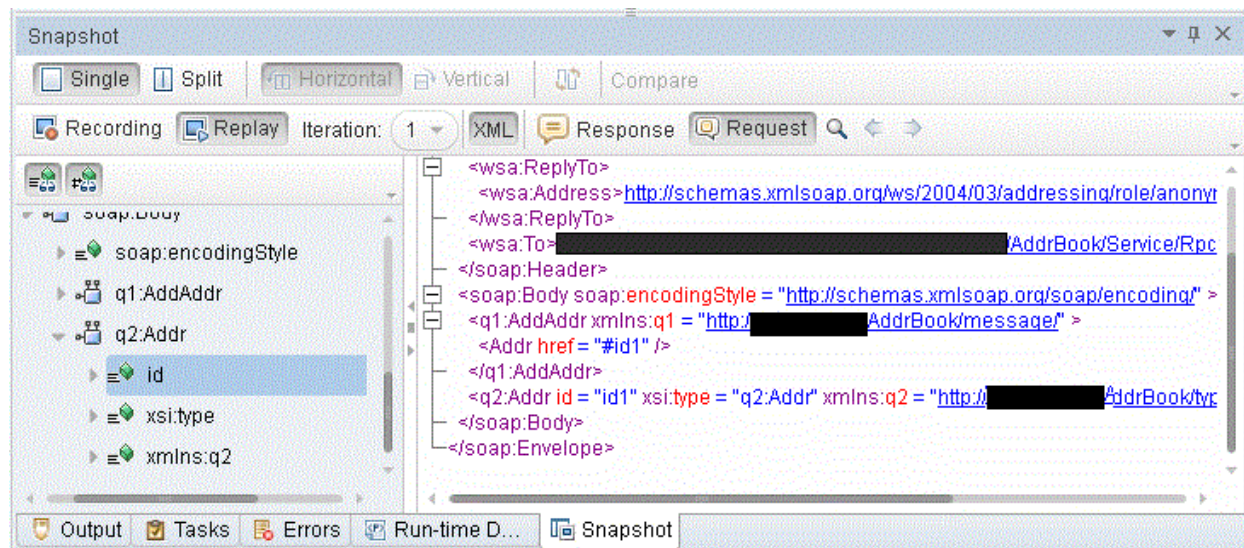
- For details on how to work with the Snapshot pane, see ["Work with Snapshots" on page 279](#).
- For details on the Snapshot pane UI, see ["Snapshot Pane" on page 62](#).

In addition, a new snapshot has been designed to show Flex data in several views:

- Raw Data
The data received from the server that has not been formatted or parsed in any way.
- Response Body
Data entity received from the server
- Request Body
Data entity sent to the server
- Headers

If the Response Body and the Request Body are in XML format, the data can be displayed as:



- Text
- Hex
- XML



Serialize Flex Scripts

External Java Serializer

You can use the Java classes from the Flex server to serialize AMF messages in your script. This process has been simplified so that you need to include the application JAR files only if the AMF objects implement an externalizable interface.

1. In the **Recording Options > Flex > Externalizable Objects** node, select **Serialize objects using** and select **Custom Java Classes** from the drop-down menu.
2. Add the relevant files by using the **Add all classes in folder**  or **Add JAR or Zip file**  buttons. Add the following files:
 - a. **For Adobe BlazeDS or Adobe LCDS**, add the following JAR files:
 - flex-messaging-common.jar
 - flex-messaging-core.jar
 - b. Regenerate the script and note any errors. Open the recording options dialog box using the **Generation Options** button and add the necessary application JAR files.
3. Ensure that the added files exist in the same location both on the VuGen machine and on all load generators.

For details, see ["Externalizable Objects in Flex Scripts" on page 509](#).

Notes and Limitations for the Java Serializer:

- Supported JDK versions: 1.6 and earlier.
- Supported servers: Adobe BlazeDS and Adobe Livecycle DS.
- Microsoft .NET classes are not supported.
- During code generation VuGen performs a validity test of the request buffers by verifying that the buffer can be read and written using the provided jars. Failure in this validity test indicates that the classes are incompatible with VuGen.

Built-in Serializer

You can attempt to serialize externalizable objects using the built-in serializer. Ensure that you have saved all open scripts because this option may result in unexpected errors or invalid steps.

1. Save all open scripts in VuGen.
2. In the **Recording Options > Flex > Externalizable Objects** node, select **Serialize objects using** and select **LoadRunner AMF serializer**.

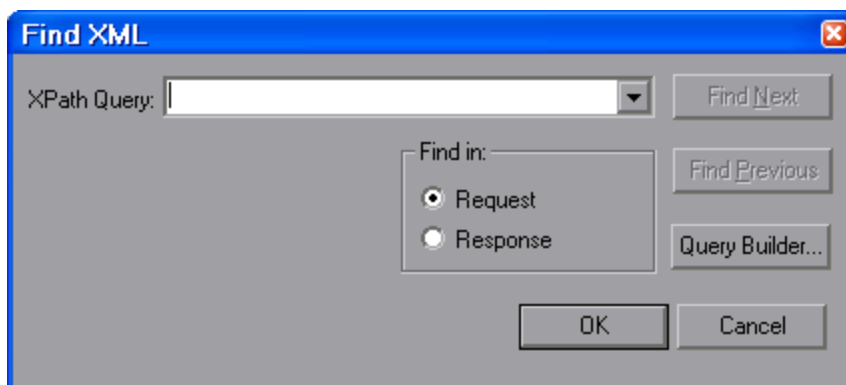
Query an XML Tree

VuGen provides a Query Builder that lets you create and execute queries on the XML.

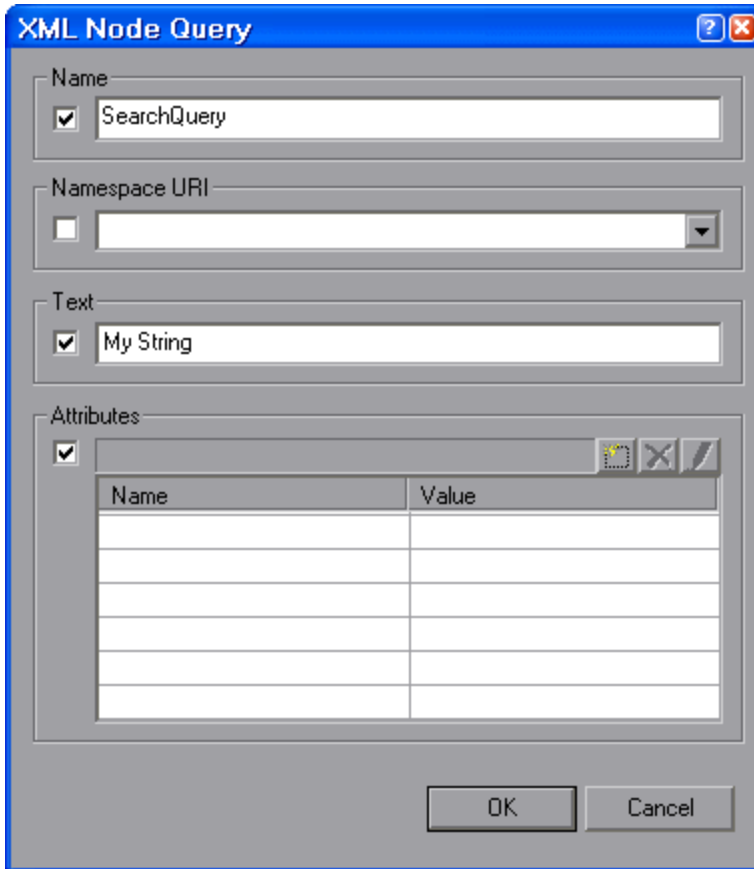
VuGen displays the XML code in an expandable tree. You can perform a query on your XML document, and search for a specific Namespace URI, value, or attribute. Note that all queries are case-sensitive.

Perform a query

1. In the Snapshot pane, select the node that you want to search. Click the **Find XML** button. The Find XML dialog box opens.



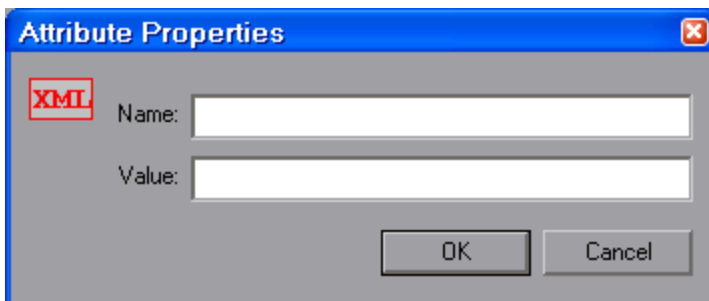
2. Select **Request** or **Response**. Enter an XPath query and click **OK**. To formulate a query, click **Query Builder** button. The XML Node Query dialog box opens.
3. Enable one or more items for searching.



The XML Node Query dialog box is used to define search criteria for XML nodes. It contains four sections: Name, Namespace URI, Text, and Attributes. Each section has a checkbox to enable it and a text input field. The Attributes section also includes a table for adding attribute names and values, and a set of icons (Add, Delete, Clear) for managing the table. The dialog has OK and Cancel buttons at the bottom.

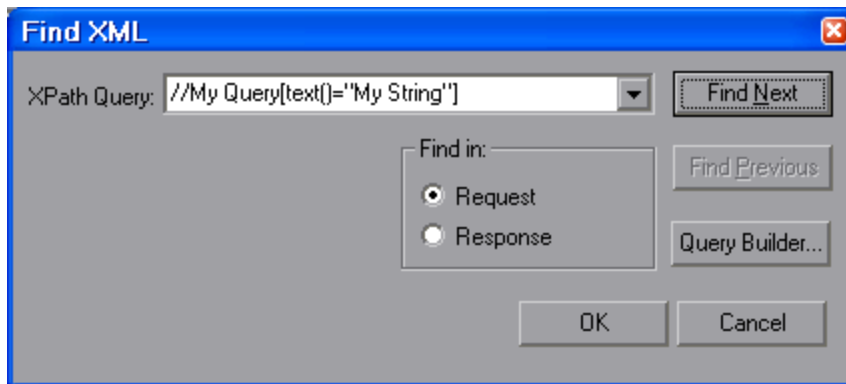
Name	Value

4. Enable the **Name** section to search for the name of a node or element.
5. Enable the **Namespace URI** section to search for a namespace.
6. Enable the **Text** section to search for the value of the element indicated in the Name box.
7. Enable the **Attributes** section to search for an attribute.
8. Enter the search text in the appropriate boxes. To add an attribute, click the **Add** button. The Attribute Properties box opens. Enter an attribute name and value. Click **OK**.



The Attribute Properties dialog box is used to define the name and value of an attribute. It has a red XML icon, a Name input field, and a Value input field. It includes OK and Cancel buttons at the bottom.

9. Click **OK** in the XML Node Query dialog box. VuGen places the text of the query in the Find XML box.



10. Click **Find Next** to begin the search.

Troubleshooting and Limitations for Flex

This section describes troubleshooting and limitations for the Flex Protocol.



Tip: For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

- A Flex script cannot be generated using an external Java Virtual Machine (JVM) version 1.4 or lower.
- The Flex protocol does not support the **Generate snapshot on error** option (**Replay > Runtime Settings > General > Miscellaneous > Error Handling**).
- Proxy recording is not supported for the Flex protocol.
- If your script contains more than one **flex_RTMP_tunneled_connect** step, with the same gateway parameter, you must insert a disconnect step for the previous **flex_RTMP_tunneled_connect** step before you connect again. For example:

```
("connect_step0",  
"SessionId=0",  
"Gateway=http://123.123.123.123:1935",  
...  
LAST);  
...  
Flex_rtmp_tunneled_disconnect("disconnect_step0",  
"SessionId=0")  
flex_rtmp_tunneled_connect("connect_step1",  
"SessionId=1",  
"Gateway=http://123.123.123.123:1935",
```

```
...  
LAST);  
...  
>Flex_rtmp_tunneled_disconnect("disconnect_step0",  
"SessionId=1")
```

- If a subsequent **flex_rtmp_tunneled_connect** command has the same gateway parameter as the previous **flex_trtmp_tunneled_connect** step and the **flex_rtmp_tunneled_disconnect** step is omitted, the script will pause indefinitely.

GraniteDS (Data Services)

- If you have modified the granite-config.xml, copy it to the **<Installation_folder>\dat** directory.
- When switching between BlazeDS and GraniteDS parsing (**Recording Options > Flex > Configuration**), VuGen must be restarted.
- LoadRunner cannot serialize both GraniteDS and BlazeDS/LCDS messages in the same script.
- All limitations that apply to AMF3 parsing, also apply to externalizable objects over RTMP.
- If the **Generate flex_rtmp_receive_stream_step** option is enabled, all transactions, comments, and rendezvous points that you add from the Recording toolbar are added to the script after the **flex_rtmp_receive_stream** step in your script.
- Web diagnostics is not supported for RTMP and RTMPT steps (even when the breakdown is enabled).
- You cannot replay two RTMPT steps at once.

Java Record Replay Protocol

Java Record Replay Protocol Overview

The Java Record Replay (JRR) protocol enables full VuGen functionality when recording a script on a Java application or applet. VuGen creates a script in pure Java and enhances it with Java-specific functions.

Supported Java Communication Protocols

VuGen supports a variety of different communication protocols for Java applications:

Protocol	Description
RMI	See "Working with RMI" on page 519
CORBA	See "Working with CORBA" on page 520

Protocol	Description
JACADA.	See "Working with Jacada" on page 522
JMS (Java Message Service).	Supports messaging between computers in a network
JDBC (Java Database Connectivity)	Used for defining how a client may access a database

VuGen's built-in support for the Java protocols utilizes hook files to define how different classes communicate with each other. For information on the hook file structure, see ["Hook File Structure" on page 532](#).

By default, VuGen only records client side activity in a script, which then emulates the load on the server. You must edit the hook file to change the actions recorded by VuGen. To manually edit the hook file, see ["Java Custom Filters Overview" on page 529](#).

Note: If you are recording a script that does not use one of the supported protocols (RMI, CORBA, JMS, Jacada, JDBC), you must define your own hook file otherwise your Vuser script will be empty.

For Java 8 and later: When more than one JDK/JRE is available, the JDK/JRE selection works as follows:

- For recording, the JRE is selected by the recorded application configuration.
- For code generation, the following methods are used to select the JRE. Each method is attempted in the order given, and the first to succeed is used.
 - a. Use the LR_JAVA_HOME environment variable. You create this variable if you want to use it to select the JRE.
 - b. Use the same JRE used for recording.
 - c. The code generation infrastructure searches for a java.exe using the default Windows application search algorithm. The user has no control over which JRE is used.
- For replay, you can specify a JDK using the Java Environment runtime settings.

Before Recording a JRR Script

In order to successfully record a script, you must install JDK on the VuGen machine before recording a script. JRE alone is insufficient. Ensure that the **classpath** and **path** environment variables are set according to the JDK installation instructions.

In addition, set the VuGen **Java VM** and **Classpath** runtime setting, under **Java Environment**.

To replay with a 64-bit JDK, in the Runtime Settings, specify the JDK path in **Java VM** and select the check box **Miscellaneous > Replay script with 64-bit**.

After recording, you can enhance or modify the script with standard Java code using JDK libraries or custom classes. VuGen utilizes the standard Java compiler, javac.exe, to compile the script. Once the script is successfully compiled you can incorporate it into a LoadRunner scenario or Business Process Monitor configuration.

Notes and limitations

- VuGen provides a tool that enables you to convert a script created for Web, into Java. For more information, see ["Convert a Web - HTTP/HTML Vuser Script into a Java Vuser Script" on page 675.](#)

Note: By default, Java 7 enables the Java Split Verifier. This prevents Java recording. VuGen uses the **-XX:-UseSplitVerifier** key while initializing the JVM during recording, to disable the verifier. This adaptation does not require any user intervention.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the vuser_init, Action and vuser_end actions. This applies to all functions named lr*.

Java Record Replay Protocol Recording Tips

This section gives tips and important information to consider when recording a Java Record Replay Vuser Script.

- The Java Record Replay protocol can only record 32-bit applications. When you specify an application, make sure to specify the 32-bit version.
- You can also specify a batch (.bat) file as the application to record.
- When you load an applet or application from VuGen during recording, it may take several seconds longer than if you were to load it independent of VuGen.
- Make sure that you have properly installed a JDK version on the machine running the Vusers—JRE alone is insufficient.
- Verify that the classpath and path environment variables are set according to the JDK installation instructions. Before you replay a Vuser script, verify that your environment is configured properly for the JDK and relevant Java classes. For details on how to set the Java Environment settings, see ["Java > VM Recording Options" on page 181](#) and ["Java > Classpath Recording Options" on page 182.](#)
- Ensure your code is thread-safe if you intend to run the Java Vuser script as a thread.
- Make sure you fill in the correct information in the Start Recording dialog box.

Start Recording - [JavaRecordReplay1]

Action selection:

Record into action: ★ Actions

Recording mode:

Record: Java application

Main class: ★ cad.driver.example.CADDriver

Parameters: 127.0.0.1

Settings:

Working directory: ★ C:\UzippedExecutableJAR\RunTestCAD

[Recording Options](#)

Start Recording Cancel

Note: The executable's jar file must be unzipped.

See also:

- ["VuGen Workflow" on page 113](#)
- ["Recording - Overview" on page 131](#)

Working with RMI

This section describes the elements of the Java Vuser script that are specific to RMI. VuGen provides full support for the RMI over IIOP protocol.

Depending on what you are recording, you can utilize VuGen's RMI recorder to create a script that will optimally emulate a real user for:

- Pure RMI client: Recording a client that uses native JRMP protocol for remote invocations
- RMI over IIOP client: Recording a client application that was compiled using the IIOP protocol instead of JRMP (for compatibility with CORBA servers).

RMI does not have constructs (as in CORBA). Instead, it uses Serializable Java objects. In RMI there is no specific shutdown section (unlike CORBA).

The following code example locates a naming registry and utilizes a lookup operation to obtain a specific Java object. You can then perform functions such as **set_sum**, **increment**, and **get_sum** on the object. You must import the RMI classes to access the RMI functions.



Example:

```
Import java.rmi.*;
Import java.rmi.registry.*;

public int action() throws Throwable {

    _registry = LocateRegistry.getRegistry("localhost",1099);
    counter = (Counter)_registry.lookup("Counter1");
    counter.set_sum(0);
    counter.increment();
    counter.increment();
    counter.get_sum();
    return lr.PASS;
}
```

When recording RMI Java, your script may contain several calls to **lr.deserialize**, which deserializes all of the relevant objects. The **lr.deserialize** calls are generated because the object passed to the next invocation could not be correlated to a return value from any of the previous calls. VuGen therefore records its state and calls the **lr.deserialize** function to represent these values during replay. The deserialization is done before VuGen passes the objects as parameters to invocations. For more information, see [Correlating Java Scripts - Serialization](#).

To set RMI timeouts, see ["Specifying connection timeouts" on page 547](#).



Note: To replay with a 64-bit JDK, in the Runtime Settings, specify the JDK path in **Java VM** and select the check box **Miscellaneous > Replay script with 64-bit**.

Working with CORBA

CORBA (Common Object Request Broker Architecture) is a powerful distributed application development architecture.

CORBA Application Vendor Classes

Running CORBA applications with JDK1.2 or later might load the JDK internal CORBA classes instead of the specific vendor CORBA classes. To force the virtual machine to use the vendor classes, specify the following java.exe command-line parameters:

Visigenic 3.4

```
-Dorg.omg.CORBA.ORBClass=com.visigenic.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.visigenic.vbroker.orb.
  ORBSingleton
```

Visigenic 4.0

```
-Dorg.omg.CORBA.ORBClass=com.inprise.vbroker.orb.ORB  
-Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSingleton
```

OrbixWeb 3.x

```
-Dorg.omg.CORBA.ORBClass=IE.Iona.OrbixWeb.CORBA.ORB  
-Dorg.omg.CORBA.ORBSingletonClass=IE.Iona.OrbixWeb.CORBA.  
    singletonORB
```

OrbixWeb 2000

```
-Dorg.omg.CORBA.ORBClass=com.ionacorba.art.artimpl.ORBImpl  
-Dorg.omg.CORBA.ORBSingletonClass=com.ionacorba.art.artimpl.  
    ORBSingleton
```

Note: Java 8 is not supported in CORBA.

Editing a CORBA Vuser Script

CORBA-specific scripts usually have a well-defined pattern. The first section contains the ORB initialization and configuration. The next section indicates the location of the CORBA objects. The following section consists of the server invocations on the CORBA objects. The final section includes a shutdown procedure which closes the ORB. Note that pattern is not mandatory and that each one of these sections may appear multiple times within a script.

In the following segment, the script initializes an ORB instance and performs a bind operation to obtain a CORBA object. VuGen imports all the necessary classes.



Example:

```
import org.omg.CORBA.*;  
import org.omg.CORBA.ORB.*;  
import lrapi.lr;  
  
public class Actions {  
  
    public int init() throws Throwable {  
        // Initialize Orb instance...  
        MApplet mapplet = new MApplet("http://chaos/classes/", null);  
        orb = org.omg.CORBA.ORB.init(mapplet, null);  
  
        // Bind to server...  
        grid = grid_dsi.gridHelper.bind("gridDSI", "chaos");  
        return lr.PASS;  
    }  
}
```

The `org.omg.CORBA.ORB` function makes the connection to ORB. Therefore, it should only be called once. When running multiple iterations, place this function in the **init** section.

In the following section, VuGen recorded the actions performed on a grid CORBA object.

```
public int action() throws Throwable {  
  
    grid.width();  
    grid.height();  
    grid.set(2, 4, 10);  
    grid.get(2, 4);  
    return lr.PASS;  
}
```

At the end of the session, VuGen recorded the shutdown of the ORB. The variables used throughout the entire recorded code appear after the **end** method and before the Actions class closing curly bracket.

```
public int end() throws Throwable {  
    if (lr.get_vuser_id() == -1)  
        orb.shutdown();  
    return lr.PASS;  
}  
// Variable section  
    org.omg.CORBA.ORB orb;  
    grid_dsi.grid ;  
}
```

Note: The ORB shutdown statement was customized for this product. This customization prevents a single Vuser's shutdown from shutting down all other Vusers.

Working with Jacada

The Jacada Interface Server provides an interface layer for mainframe applications. This layer separates the user interface from the application logic in order to insulate the organization from changes in standards and technologies.

Note: Java 8 is not supported in Jacada.

Recording a Jacada Vuser

VuGen records Jacada's Java thin-client. To record communication with the Jacada server through the HTML thin-client, use the Web HTTP/HTML type Vuser. For more information, see ["Web Protocols \(Generic\)" on page 691](#).

Before replay, you must also download the **clbase.jar** file from the Jacada server. All classes used by the Java Vuser must be in the classpath—either set in the machine's classpath environment variable or in the **Classpath Entries** list in the **Classpath** node of the runtime settings.

During replay, the Jacada server may return screens from the legacy system, in a different order than they appear in the recorded script. This may cause an exception in the replay. For information on how to handle these exceptions, contact HPE support.

Editing a Jacada Vuser Script

The Actions method of a Java Vuser script using Jacada, has two main parts: properties and body. Use the properties section to retrieve and set the server properties. Once you have the server properties you can connect to the Jacada server.

```
// Set system properties...
_properties = new Properties(System.getProperties());
_properties.put("com.ms.applet.enable.logging", "true");
System.setProperties(_properties);

_jacadavirtualuser = new cst.client.manager.JacadaVirtualUser();

lr.think_time(4);
_jacadavirtualuser.connectUsingPorts("localhost", 1100, "LOADTEST", "", "",
""");
```

The body of the script contains the user actions along with the exception handling blocks for the `checkFieldValue` and `checkTableCell` methods.



```
Example: try {
    _jacadavirtualuser.checkFieldValue(23, "S44452BA");
} catch(java.lang.Exception e) {
    lr.log_message(e.getMessage());
}

try {
    _jacadavirtualuser.checkTableCell(41, 0, 0, "");
} catch(java.lang.Exception e) {
    lr.log_message(e.getMessage());
}
```

The **checkField** method has two arguments: field ID number and expected value. The **checkTableCell** method has four arguments: table ID, row, column, and expected value. If there is a mismatch between the expected value and the received value, an exception is generated.

By default, the try-catch wrapper blocks are commented out. To use them in your script, remove the comment markers.

In addition to the recorded script, you can add any of the Java Vuser API functions. For a list of these functions and information on how to add them to your script, see ["Java Vuser Protocol" on page 536](#).

Working with JDBC

VuGen provides full support for the JDBC protocol which is used for defining how a client may access a database.

Note: Support for Java 8 only.

Supported Databases

The JDBC protocol supports all databases used via standard JDBC interface.

It also has custom support for non-standard usage of following databases:

- Postgres
- MySQL
- Oracle
- MSSQL
- H2

Recording JDBC Scripts

To record JDBC script use the Java Record Replay protocol.

In Recording Options, under **Recording Protocol > Recorder Options**, select **JDBC** and record as with any Java Record Replay script.

To avoid long and repetitive scripts, configure VuGen to generate loops where necessary by selecting **Recording Protocol > Code Generation Options > Search for loops**.

For more information, see about recording Java Record Replay scripts, see ["Java Record Replay Protocol Recording Tips" on page 518](#).

Limitations

- Blob and Clob class recording is not supported.
- Advanced non-standard features are not supported.

Sample Script

```
import lrapi.lr;

public class Actions{

    public int action() throws Throwable{
        _properties1 = System.getProperties();
        _properties1.put("com.hp.e.lr.java2.record.format", "XML");
        System.setProperties(_properties1);
    }
}
```

```
        _driver1 = new org.h2.Driver();
        _driver2 = org.h2.Driver.load();

        _string1 = "jdbc:h2:~/test";
        _string2 = "user";
        _string3 = lr_unmask("3fbae0ecdbfa75c3");
        _connection1 = java.sql.DriverManager.getConnection(_string1, _string2, _string3);

        _string4 = "SELECT count(1) FROM USERS";
        _preparedstatement1 = _connection5.prepareStatement(_string4);

        _resultset1 = _preparedstatement1.executeQuery();

        _boolean1 = _resultset1.next();

        _int1 = _resultset1.getInt(1);
        _connection1.close();

        return 0;
    } // end of method: public int action() throws Throwable

    public int init() throws Throwable{
        return 0;
    } // end of method: public int init() throws Throwable

    public int end() throws Throwable{
        return 0;
    } // end of method: public int end() throws Throwable

    /* ----- fields ----- */

    java.util.Properties _properties1;
    org.h2.Driver _driver1;
    org.h2.Driver _driver2;
    java.lang.String _string1;
    java.lang.String _string2;
    java.lang.String _string3;
    java.sql.Connection _connection1;

    java.lang.String _string4;
    java.sql.PreparedStatement _preparedstatement1;
    java.sql.ResultSet _resultset1;
    boolean _boolean1;
    int _int1;
}
```

Manually Insert Java Methods

You can use pre-defined Java packages within a Vuser script. The packages can be included in specific archive files or in the archive files contained in designated folders. You use the "Insert Java Function" dialog box to add the functions to your Vuser script. Packages, classes, methods, and other objects are represented by various icons in the "Insert Java Function" dialog box. For a list of icons, see ["Java Icon Reference List" on page 529](#).

You can customize the function generation settings by modifying the recording options. For more information, see ["General > Script Recording Options" on page 169](#).

To Insert Java Functions:

1. Click within your script at the desired point of insertion.
2. Click **Java Function** in the toolbar. The Insert Java Function dialog box opens. Under **Locations**, VuGen lists the paths and archives defined in the CLASSPATH environment variable.
3. To add a path, click **Select Folder** and then locate and highlight the required folder. To add an archive (**jar** or **zip**), click **Select Library** and then locate and highlight one or more archive files. When you select a folder or an archive file, VuGen lists it in the **Locations** box.
4. Repeat step 3 for each path or archive you want to add.
5. Select or clear the check boxes to the left of each item in the **Locations** list. If an item is selected, its members will be listed in the **Packages** list below.
6. Click **Packages** to close the Locations area and view the available packages.
7. Click the arrows to the left of each item in the navigator, to expand or collapse the trees.
8. Select an object and click **Insert**. VuGen inserts the object at the location of the cursor in the script. To paste all the methods of a class into your script, select the class and click **Insert**.
9. Repeat the previous step for all of the desired methods or classes, and then click **Close**.

Note: When you close the "Insert Java Function" dialog box, VuGen gives you the option to add the newly added locations to the Classpath list in the Java Environment runtime settings.

10. Modify the method parameter. If the script generation setting **DefaultValues** is set to **true**, you can use the default values inserted by VuGen. If **DefaultValues** is set to **false**, you must add parameters for all methods you insert into the script.

In addition, modify any return values. For example, if your script generated the following statement `"(String)=LavaVersion.getVersionId();"` , replace `(String)` with a string type variable.

11. Add any necessary statements to your script, such as imports or Java-specific functions, from the Function Reference (**Help > Function Reference**). For details, see ["Java Vuser Protocol" on page 536](#).
12. Save the script and run it from VuGen.

Manually Configure Script Generation Settings


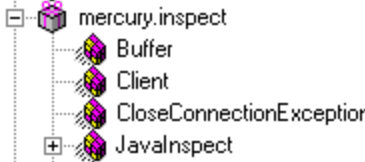
You can customize the way the navigator adds methods to your script.

To view the configuration setting, open the **jquery.ini** file in VuGen's dat folder.

```
[Display]
FullClassName=False
[Insert]
AutoTransaction=False
DefaultValues=True
CleanClassPaste=False
```

Class Name Path

The **FullClassName** option displays the complete package and class name in the Java Function navigator. This option does not affect the way the functions are added into the script—it only affects the way the classes are displayed in the navigator. By default, this option is set to false. If your packages have many classes and you are unable to view the package and class names at the same time, you should enable this option.

FullClassName enabled	FullClassName disabled
	

Automatic Transactions

The **AutoTransaction** setting creates a Vuser transaction for all methods. When you enable this option, VuGen automatically encloses all Java methods with **lr.start_transaction** and **lr.end_transaction** functions. This allows you to individually track the performance of each method. This option is disabled by default.

```
lr.start_transaction("get_host_name");
(String) = lr.get_host_name();
lr.end_transaction("get_host_name", lr.AUTO);

lr.start_transaction("isSystemClass");
(boolean) = isSystemClass ((String) "");
lr.end_transaction("isSystemClass", lr.AUTO);
```

Default Parameter Values

The **DefaultValues** setting includes default values for all methods you paste into your script. This option is enabled by default and inserts a null for all objects. If you disable this option, you must manually insert parameter values for all functions in the script. The following table illustrates the DefaultValues flag enabled and disabled.

DefaultValues enabled	DefaultValues disabled
<pre>lr.message((String) ""); lr.think_time((int) 0); lr.enable_redirection((boolean) false); lr.save_data((byte[]) null, (String) "");</pre>	<pre>lr.message((String)); lr.think_time((int)); lr.enable_redirection((boolean)); lr.save_data((byte[]), (String));</pre>

Class Pasting

The **CleanClassPaste** setting pastes a class so that it will compile cleanly: with an instance returning from the constructor, with default values as parameters, and without a need for import statements. Using this option, you will most likely be able to run your script without any further modifications. If you disable this option (default), you may need to manually define parameters and include import statements. Note that this setting is only effective when you paste an entire class into your script—not when you paste a single method.

The following segment shows the toString method pasted into the script with the CleanClassPaste option enabled.

```
_class.toString();
    // Returns: java.lang.String
```

The same method with the CleanClassPaste option disabled is pasted as follows:

```
(String) = toString();
```

The next segment shows the **NumInserter** Constructor method pasted into the script with the CleanClassPaste option enabled.

```
utils.NumInserter _numinserter = new utils.NumInserter
    ((java.lang.String) "", (java.lang.String) "", (java.lang.String)
    "" ...);
    // Returns: void
```

The same method with the CleanClassPaste option disabled is pasted as:

```
new utils.NumInserter((String) "", (String) "", (String) "", ...);
```

Compiling and Running a Script as Part of a Package

When creating a Java Record Replay or a Java Vuser script, you may need to use methods in other classes in which the class or method is protected. If you try to compile this type of script, you will receive







errors in the compilation stage indicating that the methods are inaccessible. To make sure that your script can access these methods, insert the package name containing these methods at the top of the script, just as you would do in a standard Java program— `<package_name>`. In the following example, the script defines the **my.test** package which consists of a path:

```
package my.test;
import lrapi.*;
public class Actions
{
}
```

In the above example, VuGen automatically creates the **my/test** folder hierarchy under the Vuser folder, and copies the **Actions.java** file to **my/test/Actions.java**, allowing it to compile with the relevant package. Note that the package statement must be the first line in the script, similar to Java (excluding comments).

Java Icon Reference List

The following table lists the icons that represent the various Java objects:

Icon	Item	Example
	Package	java.util
	Class	public class Hashtable extends java.util.Dictionary implements java.lang.Cloneable, java.io.Serializable
	Interface Class	public interface Enumeration
	Method	public synchronized java.util.Enumeration keys ()
	Static Method	public static synchronized java.util.TimeZone getTimeZone
	Constructor Method	public void Hashtable ()

Java Custom Filters Overview

This section describes the background information necessary to create custom Java filters. For task details, see ["Create a Custom Java Filter" on page 531](#).

When recording a Java Record Replay script, the recorder selects which methods are recorded in the Vuser script and which are left out. VuGen uses hooking to filter the methods to include in the script.

VuGen comes with built-in support for the **RMI**, **CORBA**, **JMS**, **Jacada**, and **JDBC** protocols. The built-in filters these protocols are designed to record only the server-related traffic relevant to your testing goals.

If your protocol is not supported, you can define your own custom hook file. Custom Java protocols, proprietary enhancements and extensions to the default protocols, and data abstraction all require a custom filter definition.

Note: If you are recording a script that does not use one of the supported protocols, you must define your own hook file. Otherwise, your Vuser script will be empty.

Creating a custom hook file demands planning and a good understanding of the protocols your application uses. When hooking is implemented correctly, the Vuser script should be well correlated and ready for compilation. For more details on how to select which methods and classes to hook, see "[Java Custom Filters - Determining which Elements to Include](#)" below.

When you record a method, the methods which are called from the recorded method either directly or indirectly, are not recorded.

In order to record a method, VuGen must recognize the object upon which the method is invoked, along with the method's arguments. VuGen recognizes an object if it is returned by another recorded method provided that:

- The construction method of that object is hooked.
- It is a primitive or a built-in object.
- It supports a serializable interface.

You can create a custom filter to exclude unwanted methods. When recording a Java application, your script may include calls to methods that do not affect the server, such as calls to a local utility or the GUI interface. These calls are usually not relevant to your testing goals, and it would be correct to filter them out.

Before creating a test, we recommend that you become familiar with your application and determine its primary classes and methods, so that you will know which ones to include in your recording.

If you are not familiar with your application's classes, VuGen allows you to record with a stack trace that logs all of the methods that were called by your application. In order to record with stack trace set the log level to **Detailed**.

Java Custom Filters - Determining which Elements to Include

When designing a custom filter, we recommend that you start by choosing the appropriate built-in filter as a base filter. You can then customize the filter using one of the following approaches:

- **Top Down Approach.** An approach in which you include the relevant package and exclude specific classes that are not part of the client-server activity. This is recommended if you are familiar with your application and you can identify a well-defined layer which implements all client-server activity without involving any GUI elements.

- **Bottom up Approach.** An approach in which you use the default filter and refine it by adding individual methods or classes. Use this approach if you cannot identify a well-defined layer or if you are not familiar with your application. Do not add all AUT packages and then try to remove extra component one by one.

The following section provides guidelines on when to include or exclude elements.

- If, as a result of your including a class, your script has many unrelated method calls, try modifying the filter to exclude the irrelevant methods.
- If you identify a non-client/server call in your script, exclude its method in the filter.
- During recording, VuGen may detect an unknown input argument, for example, an argument whose construction it had never encountered before. If this argument supports serialization, VuGen serializes it by saving it to a file in a special format. During replay, VuGen reconstructs the argument by deserializing it.
- VuGen serializes objects passed as arguments that were not included by the filter. We recommend that you include this object in the filter in order to track its construction and activity instead of using it in its serialized form. You can identify serialized objects in the script by searching for calls to the **Ir.deserialize()** method in your script. For more information see [Correlating Java Scripts - Serialization](#).
- Exclude all activity which involves GUI elements.
- Add classes for utilities that may be required for the script to be compiled.

Create a Custom Java Filter

This task describes how to create a custom Java filter. For background information, see "[Java Custom Filters Overview](#)" on page 529.

For details of the hook file structure, see "[Hook File Structure](#)" on the next page.

When preparing a script, you may need to customize the filter several times in order to achieve the optimal filter. An optimal filter records the relevant methods without introducing a large number of irrelevant calls to the script.

Note: If you plan to add manual code to your script such as control flow or message statements, make sure to do so after you have a functional script that runs inside VuGen. The reason for this, is that if you re-record a script after modifying the filters, it will overwrite all manual changes.

Define a Custom Hook File

1. Create a **user.hooks** file in the VuGen installation classes folder, typically **C:\Program Files (x86)\HPE\Virtual User Generator\classes**. VuGen automatically searches for this file when recording. For structural details about the user.hook file, see "[Hook File Structure](#)" on the next page.
2. Open the Recording Options dialog box (Ctrl+F7) and select the **Log Options** node. Select the Log Level to **Detailed**.
3. Record your application. Click **Start Record** (Ctrl + R) to begin and **Stop** (Ctrl + F5) to end.

4. View the script's steps. If you can determine the business logic from the steps and apply correlation, you may not need to create custom filters. If, however, the script is very long or hard to maintain and correlate, you should customize the script's filter.
5. Try to identify the high-level method in the call that captures or wraps one or more client server calls. You can do this by opening the AUT source files (if they are available) or by viewing a Stack Trace of the script.
6. Set the filter to include the relevant methods. For more information, see ["Java Custom Filters - Determining which Elements to Include" on page 530](#).
7. Record the application again. You should always rerecord the application after modifying the filter.
8. Repeat steps 4 through 7 until you get a simple script which can be maintained and correlated.
9. Correlate the script. In order for your test to run properly, you may need to insert a correlation to capture a value and use it at a later point in the script. For more information about the built-in correlation mechanism, see [Correlating Java Scripts](#).

Note: Do not modify any of the other .hooks file as it might damage the VuGen recorder.

Adding custom hooks to the default recorder is a complicated task and should be considered thoroughly as it has both functional and performance consequences.

Caution: Incorrect hooking definitions can lead to incorrect scripts, slow recording, and application freeze-up.

Hook File Structure

The following section describes the structure of a typical .hooks file, that you use when creating a custom filter. For details, see ["Create a Custom Java Filter" on the previous page](#).

```
[hook-Name]
class      = MyPackage.MyClass
method     = MyMethod
signature  = ()V
ignore_cl  =
ignore_mtd =
ignore_tree =
cb_class   = mercury.ProtocolSupport
cb_mtd     =
general_cb = true
deep_mode  = soft | hard
make_methods_public = true | false
lock       = true | false
```

The hook files are structured as .ini files where each section represents a hook definition. Regular expressions are supported in some of the entries. Any entry that uses regular expression must start with a '!.



Note: When you use a filter such as `!.*` then the `!` indicates the beginning of a RegExp—not a Regexp negation.

hook-Name

Specifies the name of this section in the hooks file. Hook-Name must be unique across all hooks files. A good practice is to give the fully qualified class name and method. For example:



Example: `[javax.jms.Queue.getQueueName]`

class

A fully qualified class name. Regular expression can be used to include several classes from the same package, a whole package, several packages, or any class that matches a name. The following example filters for any class that starts with `javax.jms` and is followed by at least one character.



Example: `class = !javax\jms\.*`

method

The simple name of the method to include. Regular expressions can be used to include more than one method from the class. For example:



Example: `method = getQueueName`

signature

The standard Java internal type signature of the method. To determine the signature of a method, run the command `javap -s class-name` where `class name` is the fully qualified name of the class. Regular expressions can be used to include several methods with the same name, but with different arguments. For example:

For example:

`signature = !.*` will match any possible signature, thus causing any method in this class to be recorded into the script regardless of signature.

`signature = !\ (Lorg/omg/CORBA/ORB; \) .*` will match any signature starting with `(Lorg/omg/CORBA/ORB;)`.

ignore_cl (optional)

A specific class to ignore from the classes that match this hook. This can be a list of comma separated

class names. Each item in the list can contain a regular expression. If an item in the list contains a regular expression, prepend a '!' to the class name. For example:



Example: `ignore_cl = !com.hp.jms.Queue,!com\hp\..*`

ignore_mtd (optional)

A specific method to ignore. When the loaded class method matches this hook definition, this method will not be hooked. The method name must be the simple method name followed by the signature (as explained above). To ignore multiple methods, list them in a comma separated list. To use a regular expression, prepend a '!' to the method name. For example:



Example: `ignore_cl = open, close`

ignore_tree (optional)

A specific tree to ignore. When the name of the class matches the ignore tree expression, any class that inherits from it will not be hooked, if it matches this hooks definition. To ignore multiple trees, list them in a comma separated list. To use a regular expression, prepend a '!' to the class name. This option is relevant only for hooks that are defined as deep.

cb_class

The callback class that gets the call from the hooked method. It should always be set to **mercury.ProtocolSupport**. The built-in hook definitions may use other classes, such as `mercury.jms.JMSSupport`.

cb_mtd (optional)

A method in the callback class that gets the call from the hooked method. If omitted, it uses the default, **general_rec_func**. For cases where you just need to lock the subtree of calls, use **general_func** instead.

general_cb

The general callback method. This value should always be set to **true**.

deep_mode

Deep mode refers to classes and interfaces that inherit or implement the class or interface that the hook is listed for. The inherited classes will be hooked according to the type of hook: **Hard**, **Soft**, or **Off**.

- **Hard.** Hooks the current class and any class that inherits from it. If regular expressions exist, they are matched against every class that inherits from the class in the hook definition. Interface inheritance is treated the same as class inheritance.
- **Soft.** Hooks the current class and any class that inherits from it, only if the methods are overridden in the inheriting class. If the hook lists an interface, then if a class implements this interface those methods will be hooked. If they exist in classes that directly inherit from that class they will also be hooked. However, if the hook lists an interface and a class implements a second interface that inherits

from this interface, the class will not be hooked.

Note: Regular expressions are not inherited but converted to actual methods.

- **Off.** Only the class listed in the hook definition and the direct inheriting class will be hooked. If the hook lists an interface, only classes that directly implement it will be hooked.

make_methods_public (optional)

Any method that matches the hook definition will be converted to public. This is useful for custom hooks or for locking a sub tree of calls from a non-public method.

Note that this applies only during record. During replay, the method will use the original access flags. In the case of non-public methods, it will throw `java.lang.VerifyError`.

lock (optional)

When set to **true**(default), it locks the sub tree and prevents the calling of any method originating from the original method.

When set to **false**, it will unlock the sub tree, record any method originating from the current method (if it is hooked), and invoke the callback.

Troubleshooting and Limitations - Java Record Replay and Java Vuser

This section describes troubleshooting and limitations for the Java Vuser protocol and the Java Record Replay protocol.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

- When recording on Internet Explorer 8 using the Java protocol, you must first close all instances of Internet Explorer before LoadRunner opens an Explorer instance for the record session.
- (Java Record Replay Protocol only) Recording of JMS applications requires JDK version 1.7 or 1.6u32 and lower.
- Due to a restriction in JVM architecture, a method cannot exceed 64 KB.
- When you run a Java script, the replay status may be "Script Not Run" and some errors may appear in the `mdrv.log` file. However, due to Java internal architecture, these errors may not be included in the VuGen Output and Errors panes. This occurs when VuGen fails to initialize a Java Vuser, and JVM then terminates the replay process.

Workaround: Look for errors directly in the `mdrv.log` file. If the entry in the log is due to a memory-related issue, try using different memory options for Java in the runtime settings.

Specifying connection timeouts

To set timeouts, add Java code to set the properties.

Example of setting RMI timeouts to 5 seconds:

```
import lrapi.lr;
public class Actions {
    public int init() throws Throwable {
        return 0;
    }
    public int action() throws Throwable {
        java.util.Properties properties=System.getProperties();

        properties.put("sun.rmi.transport.tcp.responseTimeout", 5000);
        properties.put("sun.rmi.transport.tcp.readTimeout", 5000);
        properties.put("sun.rmi.transport.connectionTimeout", 5000);
        properties.put("sun.rmi.transport.handshakeTimeout", 5000);

        properties.put("user.script", "");
        System.setProperties(properties);
        java.lang.String var_0="rmi://example.com/RmiServer";
        RmiServerIntf var_1=(RmiServerIntf)java.rmi.Naming.lookup(var_0);
        java.lang.String var_2=var_1.getMessage();
        return 0;
    }
    public int end() throws Throwable {
        return 0;
    }
}
```

To set this timeout to all TCP connections:

```
properties.put("sun.net.client.defaultReadTimeout", 5000);
properties.put("sun.net.client.defaultConnectTimeout", 5000);
```

Java Vuser Protocol

Manually Programming Java Scripts - Overview

To prepare Vuser scripts using Java code, use the **Java** type Vusers. This Vuser type supports Java on a protocol level. The Vuser script is compiled by a Java compiler and supports all of the standard Java conventions. For example, you can insert a comment by preceding the text with two forward slashes "//".

The first step in creating a Java compatible Vuser script, is to create a new Vuser script template of the type **Java Vuser**. Then, you program or paste the desired Java code into the script template. You can add Java Vuser functions to enhance the script and parameterize the arguments to use different values during iterations.

Set the VuGen **Java VM** and **Classpath** run-time settings under **Java Environment**.

To replay with a 64-bit JDK, in the Runtime Settings, specify the JDK path in **Java VM** and select the check box **Miscellaneous > Replay script with 64-bit**.

The Java Vuser script runs as a scalable multi-threaded application. If you include a custom class in your script, make sure that the code is thread-safe. Code that is not thread-safe may cause inaccurate results. For code that is not thread-safe, run the Java Vusers as processes. This creates a separate Java Virtual Machine for each process, resulting in a script that is less scalable.

After you prepare a script, run it as a standalone test from VuGen. A Java compiler (javac), checks it for errors and compiles the script.

After you create a script, you integrate it into your environment: a LoadRunner scenario, Performance Center load test, or Business Process Monitor configuration.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the `vuser_init`, `Action` and `vuser_end` actions. This applies to all functions named `lr*`.

Java Protocol Programming Tips

When programming a Java Vuser script, you can paste ready-made code segments into scripts or import ready-made classes in order to invoke their methods. If Vusers need to run as threads under the Controller (for scalability reasons), you need to make sure that all of the imported code is thread-safe.

Thread-safety is often difficult to detect. A Java Vuser may run flawlessly under VuGen and under the Controller with a limited number of Vusers. However, problems may then occur with a large number of Vusers. Code that is not thread-safe is usually the result of static class member usage as shown in the following example:

```
import lrapi.*;
public class Actions
{
    private static int iteration_counter = 0;
    public int init() {
        return 0;
    }
    public int action() {
        iteration_counter++;
        return 0;
    }
    public int end() {
        lr.message("Number of Vuser iterations: "+iteration_counter);
        return 0;
    }
}
```

When you run one Vuser, the **iteration_counter** member determines the number of iterations that were executed. When multiple Vusers run together as threads on a single virtual machine, the static class member **iteration_counter** is shared by all threads, resulting in an incorrect counting. The total number of all Vusers iterations is counted.

If code is known to be non thread-safe and you still want to import it into your script, you can run the Vusers as processes. For more information on running Vusers as threads or processes, see ["Runtime Settings Overview" on page 283](#).

When you run a basic Java Vuser script, it usually consists of a single thread—the main thread. Only the main thread can access the Java Vuser API. If a Java Vuser spawns secondary worker threads, using the Java API may cause unpredictable results. Therefore, we recommend that you use the Java Vuser API only in the main thread. Note that this limitation also affects the **lr.enable_redirection** function.

The following example illustrates where the LR API may and may not be used. The first log message in the execution log indicates that the value of flag is false. The virtual machine then spawns a new thread `set_thread`. This thread runs and sets flag to true, but will not issue a message to the log, even though the call to `lr.message` exists. The final log message indicates that the code inside the thread was executed and that flag was set to true.

```
boolean flag = false;
public int action() {
    lr.message("Flag value: "+flag);
    Thread set_thread = new Thread(new Runnable(){
        public void run() {
            lr.message("LR-API NOT working!");
            try {Thread.sleep(1000);} catch(Exception e) {}
            flag = true;
        }
    });
    set_thread.start();
    try {Thread.sleep(3000);} catch(Exception e) {}
    lr.message("Flag value: "+flag);
    return 0;
}
```

Running Java Vuser Scripts

Java Vuser scripts differ from C Vuser scripts in that they are first compiled and then executed; C Vuser scripts are interpreted. VuGen locates the **javac** compiler from within the JDK installation and compiles the Java code inside the script. This stage is indicated by the **Compiling...** status message in the bottom of the VuGen window. If errors occur during compilation, they are listed in the execution log. To go to the code in your script that caused the error, double-click on the error message containing the line number of the error. Fix the error and run the script again.

If the compilation succeeds, the status message **Compiling...** changes to **Running...** and VuGen begins to execute the script. When you run the script again, VuGen runs the script without recompiling it, provided that no changes were made to the script. To debug your script further, you can use breakpoints and animated run type execution using the step option.


Note: If you are making calls to JNDI extensions within your script, you may encounter problems trying to run your Vusers as **threads**. This happens because JNDI requires each thread to have its own context class loader. In order to run as threads, instruct each Vuser to run with its own context class loader, by adding the following line to the beginning of the **init** section:

```
DummyClassLoader.setContextClassLoader();
```

Editing and Running Scripts in Eclipse

You can use supported versions of Eclipse to take advantage of additional tools that enable you to view, edit, and debug your Java Vuser (Java Record Replay, and Java over HTTP) scripts. You can add breakpoints, view variable values, add references, and edit the script using IntelliSense. You can also run the script in a step-by-step mode for debugging.

When you save your script, VuGen creates java source files in your script's folder. You can open the solution file in Eclipse and view all of its components in the Projects Explorer.

To open the Vuser script in Eclipse, click the **Open Script in Eclipse** button  on the VuGen toolbar. If this is your first time using Eclipse from within VuGen, it will automatically install the Eclipse plugin.

Note: Before opening a script in Eclipse, you need to set the location of the Eclipse IDE in the **Java** node of the VuGen's Scripting options. If you do not set this value, VuGen prompts you to select its location. For details, see ["Scripting Options Tab" on page 96](#).

An additional toolbar menu provides access to common VuGen commands, such as Runtime Settings, Parameter List, Run, and Stop.

VuGen also has an add-in for Eclipse developers that allows you to create JUnit tests that can be called directly from the testing application, such as the LoadRunner Controller, without having to open them in VuGen. The add-ins are located in the **DVD/Additional Components** folder. For details, see ["Additional Components" on page 869](#).

For more information, see ["Creating Vuser Scripts or Unit Tests in Visual Studio or Eclipse" on page 839](#)


Opening Java Vuser Scripts in Eclipse

Eclipse provides you with additional tools to view, edit, and debug your Java Vuser (such as Java Record Replay and Java over HTTP) scripts. You can add breakpoints, view variable values, add references, and edit the script in the Eclipse editor using IntelliSense.

The VuGen and Eclipse integration allows you to configure the script as you would in VuGen, from supported versions of the Eclipse IDE. A **Vuser** menu added to the Eclipse IDE, provides access to the Parameter List, runtime settings, run/stop control, and scenario creation.

Note: For details on supported Eclipse versions, see the [System Requirements](#).

To open the Vuser script in Eclipse:

1. Make sure you have Eclipse 4.2 or higher on your machine, running with JDK 1.7 or later.
2. Set the location of the Eclipse IDE in the **Scripting > Java** node in VuGen's Options dialog box. For details, see ["Scripting Options Tab" on page 96](#).
3. Create a Java script (Java Vuser, Java Record Replay, Java over HTTP, and so forth).
4. Click the **Open Script in Eclipse** button  on the VuGen toolbar. If this is your first time using Eclipse from within VuGen, it will automatically install the VuGen Eclipse plugin.
5. Double-click the appropriate section, such as **Actions.java**, to edit the code.
6. Use the **Vuser** menu to define parameters, configure runtime settings, and run the script directly from the Eclipse IDE.

Compiling and Running a Script as Part of a Package

When creating a Java Record Replay or a Java Vuser script, you may need to use methods in other classes in which the class or method is protected. If you try to compile this type of script, you will receive errors in the compilation stage indicating that the methods are inaccessible. To make sure that your script can access these methods, insert the package name containing these methods at the top of the script, just as you would do in a standard Java program— `<package_name>`. In the following example, the script defines the **my.test** package which consists of a path:

```
package my.test;
import lrapi.*;
public class Actions
{
}
}
```

In the above example, VuGen automatically creates the **my/test** folder hierarchy under the Vuser folder, and copies the **Actions.java** file to **my/test/Actions.java**, allowing it to compile with the relevant package. Note that the package statement must be the first line in the script, similar to Java (excluding comments).

Manually Create a Java Script

This task describes how to manually create and edit a custom Java script.

1. **Create a new script**
 - a. Open VuGen.
 - b. Select **File > New** or click the **New** button. The New Virtual User dialog box opens.
 - c. Select **Custom > Java Vuser** from the Select Vuser type list, and click **OK**. VuGen displays a blank Java Vuser script.
 - d. Click the **Actions** section in the left frame to display the **Actions** class.

2. Insert your code into the script

After generating an empty template, you can insert the desired Java code. When working with this type of Vuser script, you place all your code in the Actions class. To view the Actions class, click **Actions** in the left pane. VuGen displays its contents in the right pane.

```
import lrapi.*;
public class Actions
{
    public int init() {
        return 0;
    }
    public int action() {
        return 0;
    }
    public int end() {
        return 0;
    }
}
```

The Actions class contains three methods: init, action, and end. The following table shows what to include in each method and when each method is executed.

Script method	Used to emulate...	Is executed when...
init	a login to a server	the Vuser is initialized (loaded)
action	client activity	the Vuser is in "Running" status
end	a log off procedure	the Vuser finishes or is stopped

Init Method

Place all the login procedures and one-time configuration settings in the init method. The init method is only executed once—when the Vuser begins running the script. The following sample init method initializes an applet. Make sure to import the **org.omg.CORBA.ORB** function into this section, so that it will not be repeated for each iteration.

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;
// Public function: init
public int init() throws Throwable {
    // Initialize Orb instance...
    MApplet mapplet = new MApplet("http://chaos/classes/", null);
    orb = org.omg.CORBA.ORB.init(mapplet, null);
    ...
}
```

Action Method

Place all Vuser actions in the action method. The action method is executed according to the number of iterations you set in the runtime settings. For more information on the iteration settings, see ["Runtime Settings Overview" on page 283](#). The following sample action method retrieves and prints the Vuser ID.

```
public int action() {  
    lr.message("vuser: " + lr.get_vuser_id() + " xxx");  
    return 0;  
}
```

End Method

In the **end** method, place the code you want the Vuser to execute at the end of the script, such as logging off from a server, cleaning up the environment, and so forth.

The end method is only executed once—when the Vuser finishes running the script. In the following example, the end method closes and prints the end message to the execution log.

```
public int end() {  
    lr.message("End");  
    return 0;  
}
```

3. Insert additional API functions

VuGen provides Java-specific functions for Java Vuser scripts. These functions are all static methods of the `lrapi.lr` class.

The Java API functions are classified into several categories: Transaction, Command Line Parsing, Informational, String, Message, and Runtime functions.

For more information about each of these functions, see the [Function Reference \(Help > Function Reference\)](#). Note that when you create a new Java Vuser script, the `import lrapi.*` is already inserted into the script.

4. Insert additional Java functions

To use additional Java classes, import them at the beginning of the script as shown below.

Remember to add the classes folder or relevant jar file to the classpath. Make sure that the additional classes are thread-safe and scalable.

```
import java.io.*;  
import lrapi.*;  
public class Actions  
{  
    ...  
}
```

5. Add script enhancements

You add script enhancements such as rendezvous points, transactions, and output messages. For

more information, see ["Enhance a Java Script" below](#).

6. Set the Java environment

Before running your Java Vuser script, make sure that the environment variables, path and classpath, are properly set on all machines running Vusers:

- To compile and replay the scripts, you must have a complete JDK installation. The installation of the JRE alone is not sufficient. It is preferable not to have more than one JDK or JRE installation on a machine. If possible, uninstall all unnecessary versions. For information about supported versions, see the product's *Installation Guide*.
- The **PATH** environment variable must contain an entry for **JDK/bin**.
- For JDK 1.1.x, the **CLASSPATH** environment variable must include the **classes.zip** path, (**JDK/lib** subfolder) and all of the VuGen classes (**classes** subfolder).
- All classes used by the Java Vuser must be in the classpath—either set in the machine's **CLASSPATH** environment variable or in the **Classpath Entries** list in the Classpath node of the Runtime settings.

Enhance a Java Script

This task describes how to enhance custom Java scripts.

Inserting Transactions

You define transactions to measure the performance of the server. Each transaction measures the time it takes for the server to respond to specified requests. These requests can be short or complex tasks.

When working with LoadRunner, you can analyze the performance per transaction during and after the scenario run, using online monitor and graphs.

You can also specify a transaction status: lr.PASS or lr.FAIL. You can let the Vuser automatically determine if the transaction was successful, or you can incorporate it into a conditional loop. For example, in your code you can check for a specific return code. If the code is correct, you issue a lr.PASS status. If the code is wrong, you issue an lr.FAIL status.

Mark a transaction

1. Insert **lr.start_transaction** into the script, at the point where you want to begin measuring the timing of a task.
2. Insert **lr.end_transaction** into the script, at the point where you want to stop measuring the task. Use the transaction name as it appears in the **lr.start_transaction** function.
3. Specify the desired status for the transaction: lr.PASS or lr.FAIL.

```
public int action() {  
    for(int i=0;i<10;i++)  
    {
```

```

    lr.message("action()"+i);
    lr.start_transaction("trans1");
    lr.think_time(2);
    lr.end_transaction("trans1",lr.PASS);
}
return 0;
}

```

Inserting Rendezvous Points

To emulate heavy user load on your client/server system, you synchronize Vusers to perform a task at exactly the same moment by creating a rendezvous point. When a Vuser arrives at the rendezvous point, it is held by the Controller until all Vusers participating in the rendezvous arrive.

You designate the meeting place by inserting a rendezvous function into your Vuser script.

Insert a Rendezvous Point

- Insert an `lr.rendezvous` function into the script, at the point where you want the Vusers to perform a rendezvous.

```

public int action() {
    for(int i=0;i<10;i++)
    {
        lr.rendezvous("rendz1");
        lr.message("action()"+i);
        lr.think_time(2);
    }
    return 0;
}

```

Obtaining Vuser Information

You can add the following functions to your Vuser scripts to retrieve Vuser information:

lr.get_attrb_string	Returns a string containing command line argument values or runtime information such as the Vuser ID or the load generator name.
lr.get_group_name	Returns the name of the Vuser's group.
lr.get_host_name	Returns the name of the load generator executing the Vuser script.
lr.get_master_host_name	Returns the name of the machine running the LoadRunner Controller or Business Process Monitor.

lr.get_scenario_id	Returns the ID of the current scenario. (LoadRunner only)
lr.get_vuser_id	Returns the ID of the current Vuser. (LoadRunner only)

In the following example, the `lr.get_host_name` function retrieves the name of the computer on which the Vuser is running.

```
String my_host = lr.get_host_name();
```

For more information about the above functions, see the [Function Reference \(Help > Function Reference\)](#).

Issuing Output Messages

When you run a scenario, the Controller Output window displays information about script execution. You can include statements in a Vuser script to send error and notification messages to the Controller. The Controller displays these messages in the Output window. For example, you could insert a message that displays the current state of the client application. You can also save these messages to a file.

Note: Do not send messages from within a transaction. Doing so lengthens the transaction execution time and may skew the actual transaction results.

You can use the following message functions in your Vuser script:

lr.debug_message	Sends a debug message to the Output window.
lr.log_message	Sends a message to the Vuser log file.
lr.message	Sends a message to a the Output window.
lr.output_message	Sends a message to the log file and Output window with location information.

In the following example, **lr.message** sends a message to the output indicating the loop number:

```
for(int i=0;i<10;i++)
{
    lr.message("action()" + i);
    lr.think_time(2);
}
```

For more information about the message functions, see the [Function Reference \(Help > Function Reference\)](#).

You can instruct the Vusers to redirect the Java standard output and standard error streams to VuGen's Execution log. This is especially helpful when you need to paste existing Java code or use ready-made

classes containing **System.out** and **System.err** calls in your Vuser scripts. In the execution log, standard output messages are colored blue, while standard errors are shown in red.

The following example shows how to redirect specific messages to the standard output and standard error using `lr.enable_redirection`:

```
lr.enable_redirection(true);  
System.out.println("This is an informatory message..."); // Redirected  
System.err.println("This is an error message..."); // Redirected  
lr.enable_redirection(false);  
System.out.println("This is an informatory message..."); // Not redirected  
System.err.println("This is an error message..."); // Not redirected
```

Note: When you set **lr.enable_redirection** to **true**, it overrides all previous redirections. To restore the former redirections, set this function to **false**.

For additional information about this function, see the [Function Reference \(Help > Function Reference\)](#).

Emulating User Think Time

The time that a user waits between performing successive actions is known as the think time. Vusers use the `lr.think_time` function to emulate user think time. In the following example, the Vuser waits two seconds between loops:

```
for(int i=0;i<10;i++)  
{  
    lr.message("action()"+i);  
    lr.think_time(2);  
}
```

You can use the think time settings as they appear in the script, or a factor of these values. To configure how Vusers handle think time functions, open the runtime settings dialog box. For more information, see ["Runtime Settings Overview" on page 283](#).

For more information about the `lr.think_time` function, see the [Function Reference \(Help > Function Reference\)](#).

Handling Command Line Arguments

You can pass values to a Vuser script at runtime by specifying command line arguments when you run the script. You insert command line options after the script path and filename in the Controller or Business Process Monitor. There are three functions that allow you to read the command line arguments, and then to pass the values to a Vuser script:

lr.get_attrib_double	Retrieves double precision floating point type arguments
-----------------------------	--

lr.get_attrib_long	Retrieves long integer type arguments
lr.get_attrib_string	Retrieves character strings

Your command line should have the following format, where the arguments and their values are listed in pairs after the script name:

```
script_name - argument argument_value -argument argument_value
```

The following example shows the command line string used to repeat script1 five times on the machine pc4:

```
script1 -host pc4 -loop 5
```

For more information on the command line parsing functions, see the [Function Reference \(Help > Function Reference\)](#).

For more information on how to insert the command line options, see ["Run a Vuser Script from a Command Prompt" on page 308](#).

Troubleshooting and Limitations - Java Record Replay and Java Vuser

This section describes troubleshooting and limitations for the Java Vuser protocol and the Java Record Replay protocol.



Tip: For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

- When recording on Internet Explorer 8 using the Java protocol, you must first close all instances of Internet Explorer before LoadRunner opens an Explorer instance for the record session.
- (Java Record Replay Protocol only) Recording of JMS applications requires JDK version 1.7 or 1.6u32 and lower.
- Due to a restriction in JVM architecture, a method cannot exceed 64 KB.
- When you run a Java script, the replay status may be "Script Not Run" and some errors may appear in the mdrv.log file. However, due to Java internal architecture, these errors may not be included in the VuGen Output and Errors panes. This occurs when VuGen fails to initialize a Java Vuser, and JVM then terminates the replay process.

Workaround: Look for errors directly in the mdrv.log file. If the entry in the log is due to a memory-related issue, try using different memory options for Java in the runtime settings.

Specifying connection timeouts

To set timeouts, add Java code to set the properties.

Example of setting RMI timeouts to 5 seconds:

```
import lrapi.lr;
public class Actions {
    public int init() throws Throwable {
        return 0;
    }
    public int action() throws Throwable {
        java.util.Properties properties=System.getProperties();

        properties.put("sun.rmi.transport.tcp.responseTimeout", 5000);
        properties.put("sun.rmi.transport.tcp.readTimeout", 5000);
        properties.put("sun.rmi.transport.connectionTimeout", 5000);
        properties.put("sun.rmi.transport.handshakeTimeout", 5000);

        properties.put("user.script", "");
        System.setProperties(properties);
        java.lang.String var_0="rmi://example.com/RmiServer";
        RmiServerIntf var_1=(RmiServerIntf)java.rmi.Naming.lookup(var_0);
        java.lang.String var_2=var_1.getMessage();
        return 0;
    }
    public int end() throws Throwable {
        return 0;
    }
}
```

To set this timeout to all TCP connections:

```
properties.put("sun.net.client.defaultReadTimeout", 5000);
properties.put("sun.net.client.defaultConnectTimeout", 5000);
```

Java over HTTP Protocol

Java over HTTP Protocol Overview

The Java over HTTP protocol is designed to record Java-based applications and applets. It produces a Java language script using web functions. This protocol is distinguished from other Java protocols in that it can record and replay Java remote calls over HTTP.

Set the VuGen **Java VM** and **Classpath** run-time settings under **Java Environment**.

Note: Java over HTTP supports asymmetric Java object traffic. This means that object serialization traffic is recognized even when it is on only one side of the communication. This occurs when the request is serialization and the response is plain HTTP, or vice versa.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

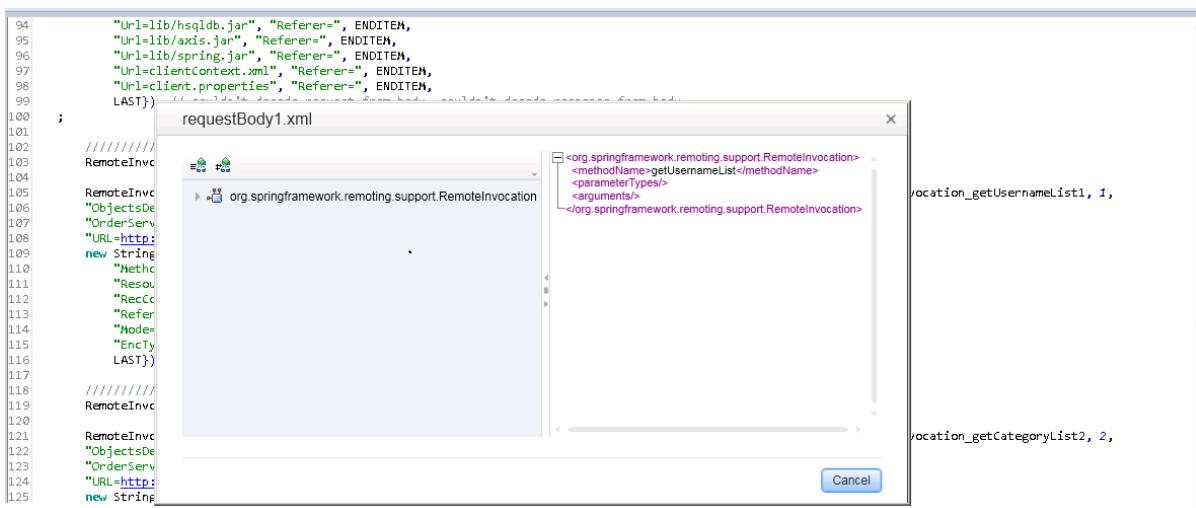
- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the vuser_init, Action and vuser_end actions. This applies to all functions named lr*.

Viewing Responses and Requests in XML Format

Note: This topic applies to the Java over HTTP protocol only.

For each request and response, you can view the corresponding XML that represents the binary java object during the recording phase.

1. Locate the target request or response section in the code. Right-click the commented **RequestBodyX.xml** or **ResponseBodyX.xml**.
2. Select **View XML**. The XML is displayed in a separate window.



Record with Java over HTTP

To record with Java over HTTP, you must specify which .jar files to use in order to deserialize the recorded data.

This topic describes how to locate the relevant .jar files and add them to the classpath.

Recording Java Applets

If your application uses Java Applets, you need to find the relevant .jar files and enable them in the classpath.

1. Clear the JAR cache by selecting **Control Panel > Java > General Tab > Temporary Internet Files > Settings > Delete Files**.

2. Open your application and perform a few business processes to repopulate the JAR cache with .jar files from your application. When you are finished, close your application.
3. Select **Control Panel > Java > General Tab > Temporary Internet Files > View**. This lists the JAR cache and should contain only the .jar files used by your application.
4. Download the files. Try the options below in the order in which they appear. When you succeed, proceed to the next step to add the .jar files to the classpath.
 - a. Option 1: For each .jar file, go to the listed URL and download the file. If you cannot download one or more of the .jar files, continue with the next option.
 - b. Option 2: Clear the cache again by selecting **Control Panel > Java > General Tab > Temporary Internet Files > Settings > Delete Files**. Open your application again and perform a few business processes. Do not close your application. Open the Java Console. There should be a message for each .jar file telling you the location it is stored in a temporary file on your computer. The files are usually hashed and don't have .jar extensions. Change the name (including changing each extension to .jar) and copy the file to a known location.
 - c. Option 3: If the files don't show up in the Java console, locate the temporary folder as listed in **Control Panel > Java > General Tab > Temporary Internet Files > Settings > Location**. Open the specified location and rename all the files in the sub-folders to .jar. Do not rename all the files in the main folder.
5. Add the .jar files to the classpath in the **Recording Options > Java Environment Settings > Classpath** node. For more information, see ["Java > Classpath Recording Options" on page 182](#).

Recording Local Java Applications

If you are recording a local Java application (not an applet), all of the .jar files already exist on your computer.

1. Look in the batch file that launched the application. All of the .jar files that are referenced should be added to the classpath.
2. If you cannot locate or understand the batch file, add all of the .jar files from the application folder and sub-folders to the classpath.
3. Add the .jar files to the classpath in the **Recording Options > Java Environment Settings > Classpath** node. For more information, see ["Java > Classpath Recording Options" on page 182](#).

Handling custom serialization and encryption of serialized objects

If the application being recorded uses an application-specific serialization, VuGen is not able to serialized objects. For example, this can happen if custom encryption is applied to serialized objects. To enable recording of such applications:

1. Implement Java interface **SerializationProviderInterface**
2. Add the jar with the implementation to the classpath in **Recording Options -> Classpath**
3. Specify the implementation class name in **Recording Options -> Java VM -> Custom SerializationProviderInterface implementation**

Correlating Java over HTTP

Automatic correlation is not supported in Java protocols, but you can manipulate script objects like any Java object.

For example: Given this business logic class:

```
public class RmiMessage implements Serializable {  
    List<Integer> integers;  
  
    public RmiMessage(List<Integer> myIntegers) {  
        this.integers = myIntegers;  
    }  
  
    public void setIntegers(List<Integer> myIntegers) {  
        this.integers = myIntegers;  
    }  
}
```

And given the following code in Action.java:

```
String _string9 = "com.hpe.lr.testing.rmi.RmiMessage __CURRENT_OBJECT = {"  
    + "java.util.List myIntegers = {"  
        + "super = {"  
            + "super = {"  
            + "}"  
            + "int modCount = #0#"  
        + "}"  
        + "java.lang.Object a[] = {"  
            + "java.lang.Integer a[0] = {"  
                + "super = {"  
                + "}"  
                + "int value = #4#"  
            + "}"  
        + "}"  
    + "}"  
}
```


The files that follow the format RequestBodyX contain the request data. The files that follow the format ResponseBodyX contain the response data.

To compare the record and replay data for the purposes of debugging, compare the files with identical names from the recording and replay phases. For example, compare the RequestBody1 file from the main folder (recording phase) to the RequestBody1 file from the replay folder.

Normally, the files should be identical. Cases where the files are not identical may indicate problems in the script.

3. Remove arguments before load testing

Return to the Java VM node and the items you added to the Additional VM Parameters field.

Insert Parameters into Java over HTTP Scripts

Parameter functions can be added for each response or request body text in a specific location. This location is indicated by a blank line, usually one to two lines below the start of the response or request body. In the example below, parameter functions can be added to the blank lines in each requestBody section.

```

//////////////////////////////// requestBody2.xml //////////////////////////////////
RemoteInvocation RemoteInvocation_getUsernameList2 =
    (RemoteInvocation) JavaHTTP.readObject(RemoteInvocationBA0);
//INSERT PARAMETERIZATION AND CORRELATION CODE HERE
RemoteInvocationResult RemoteInvocationResult_ArrayList2 =
    (RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation_getUsernameList2, 2,
    "ObjectsDeserializerDefaultImpl",
    "OrderService-httpinvoker",
    "URL=http://Kalimanjaro.devlab.ad:8080/jpetstore/remoting/OrderService-httpinvoker",
    new String[]{
        "Method=POST",
        "Resource=0",
        "RecContentType=application/x-java-serialized-object",
        "Referer=",
        "Mode=HTML",
        "EncType=application/x-java-serialized-object",
        LAST}); // 2 is the number of the header file, record time response is at file responseBody2.xml

//////////////////////////////// requestBody3.xml //////////////////////////////////
RemoteInvocation RemoteInvocation_getCategoryList3 =
    (RemoteInvocation) JavaHTTP.readObject(RemoteInvocationBA1);
//INSERT PARAMETERIZATION AND CORRELATION CODE HERE
RemoteInvocationResult RemoteInvocationResult_ArrayList3 =
    (RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation_getCategoryList3, 3,
    "ObjectsDeserializerDefaultImpl",
    "OrderService-httpinvoker_2",
    "URL=http://Kalimanjaro.devlab.ad:8080/jpetstore/remoting/OrderService-httpinvoker",
    new String[]{
        "Method=POST",
        "Resource=0",
        "RecContentType=application/x-java-serialized-object",
        "Referer=",
        "Mode=HTML",
        "EncType=application/x-java-serialized-object",
        LAST}); // 3 is the number of the header file, record time response is at file responseBody3.xml

```

Troubleshooting and Limitations for Java over HTTP

This section describes troubleshooting and limitations for the Java over HTTP protocol.



Tip: For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

Limitations

- JDK 1.5 or higher is required.
- You cannot use the GWT DFE extension or other Java-compatible DFE extensions. This is a result of a limitation of VuGen's mdrv process, which only allows you to instantiate a single JVM.
- Lazy evaluating objects are not supported, for example hibernate in lazy mode.
- If there are stateful serialization mechanisms on the application server, this can interfere with VuGen's deserialization and result in unserialized data and unexpected errors.
- The **Remote Application via LoadRunner Proxy > Display recording toolbar on client machine** option, is not supported for the Java over HTTP protocol. For details, see ["Start Recording Dialog Box" on page 221](#).
- When you run a Java script, the replay status may be "Script Not Run" and some errors may appear in the mdrv.log file. However, due to Java internal architecture, these errors may not be included in the VuGen Output and Errors panes. This occurs when VuGen fails to initialize a Java Vuser, and JVM then terminates the replay process.

Workaround: Look for errors directly in the mdrv.log file. If the entry in the log is due to a memory-related issue, try using different memory options for Java in the runtime settings.

Disable Exception Error Checking

If you are receiving exception errors and you are sure that the error is irrelevant, VuGen allows you to disable all such error messages. To do this, select **Replay > Runtime Settings > Java VM** node. In the **Additional VM Parameters** field, and append the following string to the end of the current entry:

```
-DvalidateServerResponse=false
```

Additionally, you can change the error checking behavior of a specific step by adding a closing argument to the **sendSerialized** function in script view. For more information, see the Function Reference.

Cannot Correlate Private Object Members

When you need to correlate or parameterize data that is a private member of an object, you can use the **lrapi.lr2.fieldSetter** and **lrapi.lr2.fieldGetter** functions.

```
RemoteInvocation RemoteInvocation2 = (RemoteInvocation) JavaHTTP.readObject  
(RemoteInvocationBA0);  
    RemoteInvocation.methodName="applyToSchool";  
    Student student=RemoteInvocation.arguments[0];  
  
    Map grades=lr2.fieldGetter(student,"grades");//grades is a private member
```

```
of Student
    grades.put("Math", "95");
    lr2.fieldSetter(student, "super.name", "Tom");
    //Student class inherits the name field from Person. name field is a
string
    lr2.fieldSetter(student, "super.ID", "98764321");
    //Student class inherits the ID field from Person. ID field is an int
    RemoteInvocationResult RemoteInvocationResult_ArrayList2 =
(RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation2, 2,
    "ObjectsDeserializerDefaultImpl", ....
```

LDAP Protocol

LDAP Protocol Overview

LDAP, the Lightweight Directory Access Protocol, is a protocol used to access a folder listing. The LDAP folder is composed of many LDAP entries. Each LDAP entry is a collection of attributes with a name, called a distinguished name (DN). For more information about DN, see ["Defining Distinguished Name Entries" on the next page](#).

LDAP folder entries are arranged in a hierarchical structure that reflects political, geographic, and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states or national organizations. Below them might be entries representing people, organizational units, printers, documents, or just about anything else.

VuGen records communication over LDAP servers. It creates a Vuser script, with functions that emulate your actions. This includes logging in and out of the server, adding and deleting entries, and querying an entry.

LDAP Protocol Example Script

All LDAP functions come in pairs—one for global sessions and one where you can indicate a specific session. To apply the action to all sessions, use the version without the **ex** suffix. To apply the action to a specific session, use the version with the session identifier with the **ex** suffix. For example, **mldap_logon** logs on to the LDAP server globally, while **mldap_logon_ex** logs on to the LDAP server for a specific session.

In the following example, the user logs on to an LDAP server, ldap1. It adds an entry and then renames the OU attribute from Sales to Marketing.

```
Action()
{
    // Logon to the LDAP server
    mldap_logon("Login",
        "URL=ldap://johnsmith:tiger@ldap1:80",
```

```

        LAST);

// Add an entry for Sally R. Jones
mldap_add("LDAP Add",
    "DN=cn=Sally R. Jones,OU=Sales, DC=com",
    "Name=givenName", "Value=Sally", ENDITEM,
    "Name=initials", "Value=R", ENDITEM,
    "Name=sn", "Value=Jones", ENDITEM,
    "Name=objectClass", "Value=contact", ENDITEM,
    LAST);

// Rename Sally's OU to Marketing
mldap_rename("LDAP Rename",
    "DN=CN=Sally R. Jones,OU=Sales,DC=com",
    "NewDN=OU=Marketing",
    LAST);

// Logout from the LDAP server
mldap_logoff();
return 0;
}

```

Defining Distinguished Name Entries

The LDAP API references objects by its **distinguished name** (DN). A DN is a sequence of **relative distinguished names** (RDN) separated by commas.

An RDN is an attribute with an associated value in the form attribute=value. The attribute names are not case-sensitive. The following table lists the most common RDN attribute types.

String	Attribute Type
DC	domainComponent
CN	commonName
OU	organizationalUnitName
O	organizationName
STREET	streetAddress
L	localityName
ST	stateOrProvinceName

C	countryName
UID	userid

The following are examples of distinguished names:

DN=CN=John Smith,OU=Accounting,DC=Fabrikam,DC=COM
DN=CN=Tracy White,CN=admin,DC=corp,DC=Fabrikam,DC=COM

The following table lists reserved characters that cannot be used in an attribute value.

Character	Description
	space or # character at the beginning of a string
	space character at the end of a string
,	comma
+	plus sign
"	double quote
\	backslash
<	left angle bracket
>	right angle bracket
;	semicolon

To use a reserved character as part of an attribute value, you must precede it with an escape character, a backslash (\). If an attribute value contains other reserved characters, such as the equal sign (=) or non-UTF-8 characters, you must encode it in hexadecimal format—a backslash followed by two hex digits.

The following are examples of DN's that include escaped characters. The first example is an organizational unit name with an embedded comma; the second example is a value containing a carriage return.

DN=CN=Bitwise,OU=Docs\, Support,DC=Fabrikam,DC=COM
DN=CN=Before\0DAfter,OU=Test,DC=North America,DC=Fabrikam,DC=COM

LDAP Connection Options

Using the **ldap_login[_ex]** function, you control the way you login to the LDAP server.

When specifying the URL of the LDAP server, you specify how to connect and with what credentials.

When specifying the server's URL, use the following format:

```
ldap[s][username:[password]@][server[:port]]
```

The following table shows several examples of connections to LDAP servers.

Syntax	Description
ldap://a:b@server.com:389	Connects to the server (to 389 port) and then binds with username "a", password "b"
ldap://:@server.com	Connects to server (to default unsecured port 389) then binds anonymously with a NULL username and password
ldaps://a:@server.com	Connects to server (to default secured port 636) and then binds with username "a", password ""
ldap://@server.com, ldap://server.com	Connects to server without binding
ldap://a:b@	Binds with username "a", password "b", executing a bind on the existing session without reconnecting
ldap://:@	Binds anonymously with a NULL username and password (executes bind on existing session without reconnecting)

You can also specify LDAP modes or SSL certificates using the following optional arguments:

- **Mode.** The LDAP call mode: *Sync* or *Async*
- **Timeout.** The maximum time in seconds to search for the LDAP server
- **Version.** The version of the LDAP protocol version 1,2, or 3
- **SSLCertDir.** The path to the SSL certificates database file (cert8.db)
- **SSLKeysDir.** The path to the SSL keys database file (key3.db)
- **SSLKeyNickname.** The SSL key nickname in the keys database file
- **SSLKeyCertNickname.** The SSL key's certificate nickname in the certificates database file
- **SSLSecModule.** The path to the SSL security module file (secmod.db)
- **StartTLS.** Requires that the StartTLS extension's specific command must be issued in order to switch the connection to TLS (SSL) mode

For detailed information about these arguments, see the [Function Reference \(Help > Function Reference\)](#).

Troubleshooting and Limitations - LDAP

This section describes troubleshooting and limitations for the LDAP protocol.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

- If an LDAP version 3 script fails during replay, modify the **mldpa_logon_ex** statement to specify the version number by adding "Version=3" after "URL=.."
- Address lists are only partially supported—the script only uses the first address in the list.
- When recording LDAP scripts, the binary parameter values for certain LDAP functions (such as **mldap_add** or **mldap_modify**) are not recorded. Recording of binary parameters is part of the protocol's extended functionality and is not supported by VuGen.
- LDAP Protocol can be recorded in parallel with the Windows Socket protocol in multi-protocol recording mode. This allows you to record additional network activity generated by the LDAP application. The multi-protocol recording may result in duplicate calls to the LDAP application.
Workaround: Edit the script and manually remove the duplicate calls. Leave the the Windows Socket calls that emulate the additional networking I/O.

Mailing Service Protocols

Mailing Service Protocols Overview

The Mailing Service protocols emulate users working with email clients, viewing and sending emails. The following mailing services are supported:

- Internet Messaging (IMAP). For details, see ["IMAP Protocol Overview" below](#).
- MS Exchange (MAPI). For details, see ["MAPI Protocol Overview" on the next page](#).
- Post Office Protocol (POP3). For details, see ["POP3 Protocol Overview" on page 561](#).
- Simple Mail Transfer Protocol (SMTP). For details, see ["SMTP Protocol Overview" on page 561](#).

The mail protocols support both record and replay, with the exception of MAPI that supports only replay.

IMAP Protocol Overview

IMAP Vuser script functions record the Internet Mail Application Protocol.

Each IMAP function begins with an **imap** prefix. For detailed syntax information on these functions, see the Function Reference (**Help > Function Reference**).

In the following example, the **imap_create** function creates several new mailboxes: Products, Solutions, and FAQs.

```
Actions()
{
    imap_logon("ImapLogon",
        "URL=imap://johnd:letmein@exchange.mycompany.com",
        LAST);
    imap_create("CreateMailboxes",
        "Mailbox=Products",
```

```
        "Mailbox=Solutions",  
        "Mailbox=FAQs",  
        LAST);  
imap_logout();  
return 1;  
}
```

When recording a login step in which an IP address was specified, the script saves the IP address instead of the host name.

Note: VuGen currently only supports the IMAP LOGIN authentication method, but not the AUTHENTICATE method.

MAPI Protocol Overview

MAPI Vuser script functions generate activity to and from an MS Exchange server. Each MAPI function begins with a **mapi** prefix. You cannot record Vuser scripts using the MAPI protocol. For detailed syntax information on these functions, see the [Function Reference \(Help > Function Reference\)](#).

Note: To run MAPI scripts, you must define a mail profile on the machine running the script. For example, install Outlook Express or Microsoft Outlook, set it as the default mail client, and create a mail account.

In the following example, the **mapi_send_mail** function sends a sticky note through an MS Exchange server.

```
Actions()  
{  
    mapi_logon("Logon",  
        "ProfileName=John Smith",  
        "ProfilePass=Tiger",  
        LAST);  
    //Send a Sticky Note message  
    mapi_send_mail("SendMail",  
        "To=user1@techno.merc-int.com",  
        "Cc=user0002t@techno.merc-int.com",  
        "Subject=<GROUP>:<VUID> @ <DATE>",  
        "Type=Ipm.StickyNote",  
        "Body=Please update your profile today.",  
        LAST);  
    mapi_logout();  
    return 1;  
}
```

POP3 Protocol Overview

POP3 Vuser script functions emulate actions using the Post Office Protocol, POP3. Each function begins with a **pop3** prefix. For detailed syntax information on these functions, see the [Function Reference \(Help > Function Reference\)](#).

In the following example, the **pop3_retrieve** function retrieves five messages from the POP3 server.

```
Actions()
{
    pop3_logon("Login", "
        URL=pop3://user0004t:my_pwd@techno.merc-int.com",
        LAST);
    // List all messages on the server and receive that value
    totalMessages = pop3_list("POP3", LAST);
    // Display the received value (It is also displayed by the pop3_list function)
    lr_log_message("There are %d messages.\r\n\r\n", totalMessages);
    // Retrieve 5 messages on the server without deleting them
    pop3_retrieve("POP3", "RetrieveList=1:5", "DeleteMail=false", LAST);
    pop3_logoff();
    return 1;
}
```

Note: When recording a login step in which an IP address was specified, the script saves the IP address instead of the host name.

SMTP Protocol Overview

SMTP Vuser script functions emulate the Single Mail Transfer Protocol traffic. Each SMTP function begins with an **smtp** prefix. For detailed syntax information on these functions, see the [Function Reference \(Help > Function Reference\)](#).

In the following example, the **smtp_send_mail** function sends a mail message, through the SMTP mail server, techno.

```
Actions()
{
    smtp_logon("Logon",
        "URL=smtp://user0001t@techno.merc-int.com",
        "CommonName=Smtp Test User 0001",
        NULL);
    smtp_send_mail("SendMail",
        "To=user0002t@merc-int.com",
        "Subject=MIC Smtp: Sample Test",
        "MAILOPTIONS",
        "X-Priority: 3",
        "X-MSMail-Priority: Medium",
```

```
        "X-Mailer: Microsoft Outlook Express 5.50.400\r\n",  
        "X-MimeOLE: By Microsoft MimeOLE V5.50.00\r\n",  
        "MAILDATA",  
        "MessageText="  
            "Content-Type: text/plain;\r\n"  
            "\tcharset=\"iso-8859-1\"\r\n"  
            "Test,\r\n"  
            "MessageBlob=16384",  
        NULL);  
smtp_logout();  
return 1;  
}
```

Note: When recording a login step in which an IP address was specified, the script saves the IP address instead of the host name.

Message Protocols

MMS (Multimedia Messaging Service) Protocol Overview

Caution: This protocol is supported for replay only. Support for this protocol will be discontinued in future versions.

To use MMS protocol, you must first manually register the following dlls, using the **regsvr32** command:

- **regsvr32 MMSEngine.dll**
- **regsvr32 MM1Client.dll**

The MMS protocol is useful for sending MMS messages between mobile devices.

MMS (Multimedia Messaging Service) is an extension of the SMS protocol. An MMS message typically includes a collection of attachments. MMS usually requires a third generation (3G) network.

To receive an MMS message, a mobile phone receives an MMS notification over SMS. The SMS message can be received over various SMS protocols such as SMPP, UCP, and CIMD2. The SMS message contains a unique path to the MMS message stored in the MMSC server's database and the mobile phone uses this path to download the message from the SMSC. The current version of VuGen supports the receiving of MMS notifications over the SMPP interface.

Multimedia Messaging Service Vuser scripts support the 1.0 and 1.1 versions of the MMS protocol, as defined by OMA (Open Mobile Alliance organization). Using MMS Vusers, you can send MMS messages to the MMSC server directly over the HTTP protocol.

Run an MMS Scenario in the Controller



Caution: This protocol is supported for replay only. Support for this protocol will be discontinued in future versions.

To use MMS protocol, you must first manually register the following dlls, using the **regsvr32** command:

- **regsvr32 MMSEngine.dll**
- **regsvr32 MM1Client.dll**

An MMS (Multimedia Messaging Service) scenario requires a command line setting.

To set the MMS command line setting:

1. From the Scenario Schedule screen, click **Details**. The Group Information dialog is displayed.
2. If the Command line box is not visible, click the **More** button.
3. Add the following to the end of the Command line text: `-usingwininet yes`
4. Click **OK** to accept the Command line switch.

Mobile Protocols

Select a Recording Method for Mobile Applications

The VuGen mobile protocols expand VuGen's capability to record user activity on mobile applications, both native¹ and browser-based². With these protocols you can:

- Simulate users working on mobile devices
- Create scripts based on the recordings of mobile devices or emulators

Web HTTP/HTML

A protocol enabling you to develop scripts using mobile devices or mobile device emulators communicating with servers over HTTP. You can record network traffic into a capture file (PCAP file) and then use the PCAP file to create a Vuser script. Additionally, you can use a mobile emulator on your VuGen machine to develop your scripts. For more information, see ["Web - HTTP/HTML - Recording Methods for Mobile Applications" on page 566](#).

¹A mobile application, such as a new service, where the application resides on the device, but communicates with the server at various intervals.

²A browser-based application that has been configured for the display of the mobile device.

If your client application...	and...	You can use...
<ul style="list-style-type: none"> Communicates with the HTML/HTTP protocol. Is either a browser-based or native application. 	<ul style="list-style-type: none"> Your device is in the same network as the VuGen machine. Your device allows proxy configuration. 	"Record a Script via a Proxy" on page 216
	<ul style="list-style-type: none"> You have an existing capture file. 	"Create a Vuser Script by Analyzing a Captured Traffic File" on page 685
	<ul style="list-style-type: none"> You do not want (or cannot) record from the actual device. Your mobile OS is Android. You have a device emulator. 	"Using Emulation to Record Mobile Applications" on page 679

TruClient - Native Mobile

A protocol enabling you to record user activity in native, browser-based, or hybrid mobile applications using TruClient technology. The replay is performed on real devices, allowing you to obtain client-side measurements and test end-to-end performance. This method requires the installation of Mobile Center. For details, select the relevant version in the [Mobile Center Help](#) and see the **Performance Testing** section.

Use if your client application is either:

- Native, browser-based
- Hybrid

TruClient - Mobile Web

A protocol enabling you to record user activity in browser-based mobile applications using TruClient technology. The TruClient browser is modified to emulate the display of your mobile browser. For details, see ["TruClient - Mobile Web Protocol" on page 567](#) and ["How to Record a Script with TruClient - Mobile Web" on page 567](#).

Use if your client application is a browser-based mobile version of a Web site and supports Firefox.

SMP (SAP Mobile Platform)

A protocol enabling you to create .NET based scripts using files generated by SMP. For details, see ["SMP \(SAP Mobile Platform\) Protocol" on page 569](#).

Use if your client application is built on SMP (SAP Mobile Platform).

Speed Simulation for Mobile Vuser Scripts

Speed Simulation models the behavior of a mobile network. This enables you to test applications, while taking into consideration end-to-end response time from device to server.

Three configuration options are available, allowing you to maximize the accuracy of your simulation:

- Maximum Bandwidth
- Standard Bandwidth
- Custom Bandwidth

You set the bandwidth using the Speed Simulation Runtime settings as described below.

To access	VuGen > Runtime settings > Network > Speed Simulation
------------------	---

User interface elements are described below:

UI Element	Description
Use maximum bandwidth	Vusers run at the maximum bandwidth that is available over the network. This option is provided in cases where you do not wish to emulate a specific network. This is the default setting.
Use standard bandwidth	<ul style="list-style-type: none">• General Packet Radio Service (GPRS)• Enhanced Data rates for GSM Evolution (EDGE)• Universal Mobile Telecommunications System (UMTS)• High-Speed Downlink Packet Access (HSDPA)• High-Speed Downlink Packet Access Phase 2 (HSDPA phase 2)• High-Speed Uplink Packet Access (HSUPA) <p>Each network type has both a maximum and expected rate. The maximum rate represents the technology's best case performance rate while the expected rate more accurately reflects real time performance.</p>

UI Element	Description
Use custom bandwidth	<p>Sets a custom download and upload speed, defined in bits. You can set a single value or range for either the upload or download speed. A case where this option would be useful, is when you know the expected network speed from your cellular provider for a specific area.</p> <p>Download speed:The download speed in bits, defined as a range or single value.</p> <p>Upload speed:The upload speed in bits, defined as a range or single value.</p> <div> <p>Note:</p> <ul style="list-style-type: none"> If you select this option and either the upload speed or download speed fields are left blank, the empty value is automatically set to the value of the non-empty field. If you select this option and leave the upload speed and download speed fields blank, the default setting of maximum bandwidth will be used. </div>

Web - HTTP/HTML - Recording Methods for Mobile Applications

Note: This section describes the use of the Web - HTTP/HTML protocol for recording user activity on mobile applications. For comprehensive details about the Web - HTTP/HTML protocol, see ["Web - HTTP/HTML Protocol" on page 664](#).

The Web HTTP/HTML protocol provides the following methods for generating Vuser scripts for mobile applications:

You can use:	To do this:
A Capture File to Generate a Vuser Script	<p>You can use VuGen to analyze a capture file that was created with an external capture file utility, such as Wireshark, and then generate a Vuser script.</p> <p>For details see "Create a Vuser Script by Analyzing a Captured Traffic File" on page 685.</p>
An Emulator to Create a Vuser Script	<p>For many mobile devices, there are third party emulators that you can install on your computer. Once installed, you can use the emulator to record and generate a Vuser script for mobile applications.</p> <p>For details see "Using Emulation to Record Mobile Applications" on page 679.</p>

You can use:	To do this:
The LoadRunner Proxy to Create a Vuser Script	<p>The VuGen machine acts as a proxy server capturing all the traffic from the mobile device to the target server. After the business process has been recorded VuGen creates a script.</p> <p>For details, see:</p> <ul style="list-style-type: none">• "Recording via a Proxy - Overview" on page 214• "Record a Script via a Proxy" on page 216

TruClient - Mobile Web Protocol

TruClient - Mobile Web Protocol Overview

Based on the innovative TruClient technology, TruClient - Mobile Web enables you to test web applications designed for mobile devices.

With this protocol you can:

- Simulate various mobile browsers.
- Develop scripts that are recorded on the user level making them clear and easily maintained.

The following illustrates the workflow for using the TruClient - Mobile Web protocol:



How to Record a Script with TruClient - Mobile Web

1. Create a new script of the type **TruClient - Mobile Web**.
2. Add or import a mobile device.

Select **Tools > Mobile TruClient Device Manager > Add Mobile Device** or **Import Mobile Device**.

For details, see ["Mobile Device Dialog Box" on the next page](#).

3. Start developing the script.

Click the  button. In the Mobile Settings dialog box, select a device. If you

need to modify a setting for this recording, do so now.

For details, see ["Mobile Device Dialog Box" on the next page](#).

4. Record a business process.

For details on using TruClient's functionality, see the [TruClient Help Center](#) (select the relevant version).

Add, Remove, and Import Mobile Device Settings for TruClient - Mobile Web

Create a Custom Device Using the Mobile Device Manager

The TruClient - Mobile Web device manager is delivered with the settings for many popular mobile devices, however, you can easily add a custom device.

1. Select **Tools > Mobile TruClient Device Manger > Add Mobile Device**. This opens the Add Mobile Device dialog box.
2. Enter the name of the device you would like to add or select an existing device from the drop-down list to customize.
3. Specify the User Agent.
4. Specify the Display. For details, see "[Mobile Device Dialog Box](#)" below.
5. Click **Add**.

Remove a Mobile Device

1. Select **Tools > Mobile TruClient Device Manger > Remove Mobile Device**. This opens the Remove Mobile Device dialog box.
2. Select the device from the **Mobile Device** dropdown list and click **Remove**.

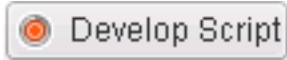
How to Import a Mobile Device Settings to Your Script

The import feature can be used to import mobile devices created in other users' scripts.

1. Open a script that contains custom device settings.
2. Select **Tools > Mobile TruClient Device Manger > Import Mobile Device**. This opens the Import Mobile Device dialog box.
3. Select the **Import Device Settings**. This opens the **Import Device Settings** dialog box.
4. Highlight the custom device and click **Import**.

Mobile Device Dialog Box

The Mobile Device dialog boxes (Mobile Settings, Add, Remove, and Import) enable you to select, add, remove, and import mobile devices for use with the TruClient - Mobile Web Protocol.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none">• Develop Script button (opens the Mobile Settings dialog box)  <ul style="list-style-type: none">• Tools > Mobile TruClient Device Manager > ...<ul style="list-style-type: none">• Add Mobile Device• Remove Mobile Device• Import Mobile Device
------------------	---

Relevant tasks	<ul style="list-style-type: none">• "Add, Remove, and Import Mobile Device Settings for TruClient - Mobile Web" on the previous page• "How to Record a Script with TruClient - Mobile Web" on page 567
-----------------------	---

UI Element	Description
Mobile Device	The mobile device under test. If you added a new device, it will appear at the end of the dropdown list.
User Agent	The header string that is sent to server to identify your mobile device. Once you have a selected a device, the default header value will appear. However, this header string can be modified.
Display	Specify the width and height of your mobile device screen. TruClient - Mobile Web will open a browser window according to the display settings.

SMP (SAP Mobile Platform) Protocol

The SMP (SAP Mobile Platform) protocol enables you to create and replay .NET based scripts using files that have been generated by SMP, formerly known as SUP. This task describes the steps to create an SMP script.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the vuser_init, Action and vuser_end actions. This applies to all functions named lr*.

Prerequisites

To create an SMP script you will need to record a business process with SMP, a platform that is provided by Sybase, an SAP company. The recording generates the following files:

- Action.cs
- Vuser_init.cs
- Vuser_end.cs

Create an SMP (SAP mobile Platform) script

1. Select **New Script and Solution > SMP (SAP Mobile Platform)**.
2. Copy the generated .cs files into the script folder.

Note: The recording mechanism for SMP scripts is disabled.

3. Save, close and reopen the script.
4. Add the location of the **SAP.Mobile.LoadRunner.dll** file using the **Replay > Runtime Settings > .NET > Shared DLLs** view.

Note: The SAP.Mobile.LoadRunner.dll has been developed and is maintained by Sybase, an SAP company.

5. Generated .cs files can include objects from external .dlls. To successfully replay the script, include a reference to these .dlls in **Runtime Settings > .NET > Shared DLLs**.
6. Replay the script. For details, see ["Debugging .NET Vuser Scripts" on page 582](#).

MQTT Protocol

MQTT is a simple, lightweight, publish/subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency or unreliable networks. MQTT is ideal for machine-to-machine (M2M) and Internet of Things (IoT) communications, as well as for mobile applications where bandwidth and battery power are at a premium. MQTT focuses on minimizing network bandwidth and device resource requirements, while attempting to ensure reliability and some degree of assurance of delivery.

Create an MQTT script

1. Create a new script by selecting **New Script and Solution > MQTT protocol**.

By default, a ready-to-use script opens using all of the available MQTT functions, as defined in the [Function Reference](#).

For general details on creating scripts, see ["Creating or Opening Vuser Scripts" on page 113](#) and ["Vuser Script Sections" on page 132](#).

2. Modify the ready-to-use script template as needed. For a complete description of each of the function calls, see the [Function Reference](#).

If you are creating a multiple-protocol script, you can integrate MQTT steps into a recorded Web - HTTP/HTML script. For details, see ["Working with multi-protocol scripts" on the next page](#).

3. Configure the runtime settings. These affect the way that a Vuser script runs.

- a. In the Solution Explorer, double-click **Runtime Settings**.

- b. In the <test name>:Runtime Settings view, configure the runtime settings as needed. To view descriptions of the individual runtime settings, hold your cursor over a runtime setting field name.

For general details, see ["Runtime Settings Overview" on page 283](#) and ["Runtime Settings Views" on page 285](#).

Working with multi-protocol scripts

You can test scenarios in which your script uses both of the following:

- MQTT to communicate between the client and the broker
- Web - HTTP/HTML to communicate between the broker and the web application

You prepare your script by recording the steps on your web application. Then you manually add MQTT steps as needed. The pre-populated MQTT script is not included when working with a multiple protocol script, but you can copy/paste from the [ready-to-use MQTT script template](#).

To enable support:

1. Create a multiple-protocol script that includes MQTT and Web - HTTP/HTML. For details, see ["Create a New Script Dialog Box" on page 128](#).
2. Record a web script.

Note: Recording a web script overwrites **globals.h**, removing the default support for MQTT steps.

3. Restore support for MQTT steps:
 - a. Open **globals.h** (located in the Solution Explorer under Extra Files).
 - b. Add **#include "MqttApi.h"** to the Include Files section.
4. Add MQTT steps to the script.

Sample multi-protocol script

```
Action()
{
    MQTT client;

    client = mqtt_create();

    mqtt_set_client_id(client, "myclientid");

    lr_start_transaction("connect_MQTT");

    mqtt_connect(client, "tcp://mymachine:1883");

    lr_end_transaction("connect_MQTT", LR_AUTO);

    lr_start_transaction("navigate_Web");

    // your recorded web code here

    lr_end_transaction("navigate_Web", LR_AUTO);

    mqtt_publish(client, "topic/subtopic", "payload", MQTT_AUTO, MQTT_DEFAULT, MQTT_
```

```
NORETAIN);

mqtt_disconnect(client);

return 0;
}
```

Ready-to-use MQTT script template

The default, single-protocol MQTT script includes actions that are pre-populated with commented out steps (shown below).

When working with multiple-protocol scripts, you first record steps on your web application. Then you add MQTT steps as needed. You can copy/paste the relevant steps from the template below.

For details on any of these functions, see the [Function Reference](#).

vuser_init

```
vuser_init()
{
    /* Declared in "globals.h" */
    client = mqtt_create();

    /* Optional connection settings */
    // mqtt_set_client_id(client, "<insert Client ID>");
    // mqtt_set_credentials(client, "<username placeholder>", "<password placeholder>");
    // mqtt_set_lwt(client, "<topic>", "<sample lwt payload>", MQTT_AUTO, MQTT_DEFAULT, MQTT_RETAIN);

    /* Optional SSL/TLS settings, applicable for SSL/TLS connections only */
    // mqtt_set_tls_certificate(client, "<path to a certificate file, e.g. cert.pem>", "<path to a private key file, e.g. key.pem>", "<password for the private key>");
    // mqtt_set_tls_parameters(client, MQTT_TLS_DEFAULT, MQTT_DEFAULT_CIPHERS);

    /* Connect to an MQTT broker */
    mqtt_connect(client, "tcp://<enter an MQTT broker host name>");

    /* Uncomment the following if implementing the "subscriber" logic */
    // mqtt_subscribe(client, "<topic>");
    return 0;
}
```

Action

```
Action()
{
    /* Uncomment the following if implementing the "subscriber" logic. Make sure to
    also uncomment subscribe/unsubscribe in vuser_init/end.c */
    //
    /* Option 1: Wait for messages, then handle their contents */
    // size_t messageCount = mqtt_await_messages(client, MQTT_DEFAULT);
    // size_t i = 0;
    // for ( ; i < messageCount; i++)

    //
    {
        // MQTT_MESSAGE m = mqtt_read_inbox(client);
        // const char* p = mqtt_get_payload(m);
        // const char* t = mqtt_get_topic(m);
        // size_t l = mqtt_get_length(m);
        // ... do something meaningful with the message
        // ... for instance, print received message information (assuming its payload is
        not binary data)
        // lr_message("received message with size %d from %s", l, t);
        // lr_message("payload %.*s", l, p);
        //
        // mqtt_free_message(m);
        // }

        /* Option 2: Wait for messages, then clear Inbox (if their handling is not needed)
        */
        // mqtt_await_messages(client, MQTT_DEFAULT);
        // mqtt_clear_inbox(client);

        /* Uncomment the following if implementing the "publisher" logic */
        // mqtt_publish(client, "<topic>", "<payload>", MQTT_AUTO, MQTT_DEFAULT, MQTT_
        RETAIN);

        return 0;
    }
}
```

vuser_end

```
vuser_end()
{
    /* Uncomment the following if implementing the "subscriber" logic */
    // mqtt_unsubscribe(client, "<topic>");

    /* Disconnect from the MQTT broker */
    mqtt_disconnect(client);
}
```

```
    return 0;  
}
```

.NET Protocol

.NET Protocol Overview

Microsoft .NET Framework provides a foundation for developers to build various types of applications such as ASP.NET, Windows Forms, Web Services, distributed applications, and applications that combine several of these models.

VuGen supports .NET as an application level protocol. VuGen allows you to create Vuser scripts that emulate users of Microsoft .NET client applications created in its .NET Framework. VuGen records all of the client actions through methods and classes, and creates Vuser scripts in C Sharp or VB .NET.

By default, the VuGen environment is configured for .NET Remoting, ADO.NET, Enterprise Services, and WCF (Windows Communication Foundation) applications. Contact Customer Support for information on how to configure VuGen to record applications created with other client-server activity.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the vuser_init, Action and vuser_end actions. This applies to all functions named lr*.

Considerations for Working with the .NET Protocol

The .NET protocol enables you to load test by replaying the application's method calls.

You can write a load test script manually, or you can generate a load test script by recording a business process.

Unlike other transport based protocols, the .NET protocol records the application method calls that are specified in the filter. Method calls that are not defined in the filter are not included in the generated script during the recording of the application.

Typically, a user is able to generate a script that can be used for load test using the default environment filter. However, for certain complex applications it may be difficult to generate a working script because the wrong method has been specified in the filter. The most difficult task of creating a load test script with the .NET protocol is resolving recording or code generation errors.

The following requirements will facilitate your ability to define the correct filter for the recording process and generate a working load test script:

- You should be familiar with .NET framework.
- You should be able to code using C# or VB.NET.
- You should be familiar with XML.
- A supported version of Visual Studio must be installed.

For details, on supported versions, see the [System Requirements](#).

- You should have an understanding of the architecture and communication techniques of the application so as to determine what functions or classes are relevant for the load test script.

The following can streamline the process of creating the correct filter:

You should have access to the application code or have some .NET reflector tools to enable you to view the decompiled code.

You should have access to the developers of the application who can help you identify the methods that are required for the load test.

For more information about .NET and the above environments, see the [MSDN Web site](#).

Viewing Data Sets and Grids

When you record a method returning a dataset, data table, or data reader action, VuGen generates a grid for displaying the data.

When working with a data reader, VuGen collects the data retrieved from each **Read** operation and converts it to the replay helper function, **DoDataRead**.

For example, after recording the following application code,

```
SqlDataReader reader = command.ExecuteReader();
while( reader.Read() )
{
    // read the values, e.g., get the string located in column 1
    string str = reader.GetString(1)
}
```

VuGen generates the following lines in the script:

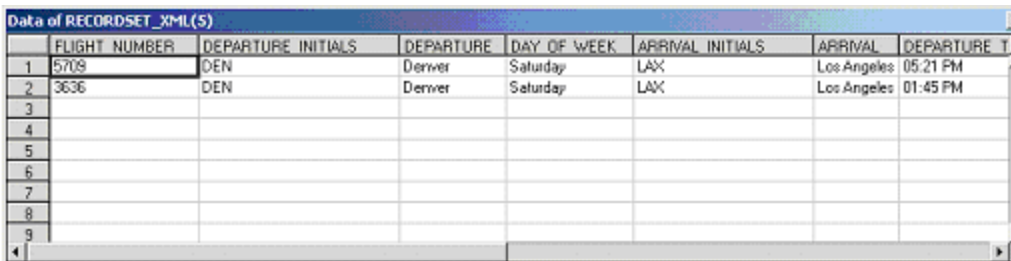
```
SqlDataReader_1 = SqlCommand_1.ExecuteReader();
LrReplayUtils.DoDataRead(SqlDataReader_1, out valueTable_1, true, 27);
```

where the two parameters indicate that during recording, the Application read all 27 available records. Therefore, during replay the script will read all available records.

In addition, VuGen generates a data grid containing all the information retrieved by the **Read** operations.

During replay you can use the output data table, containing the actual retrieved values, for correlation and verification. For more information regarding the **DoDataRead** function, see the Function Reference (**Help > Function Reference**).

When applicable, VuGen displays grid steps in the Step Navigator, and displays the associated grids in the Snapshot pane.



	FLIGHT NUMBER	DEPARTURE INITIALS	DEPARTURE	DAY OF WEEK	ARRIVAL INITIALS	ARRIVAL	DEPARTURE T
1	5709	DEN	Denver	Saturday	LAX	Los Angeles	05:21 PM
2	3636	DEN	Denver	Saturday	LAX	Los Angeles	01:45 PM
3							
4							
5							
6							
7							
8							
9							

The dataset is stored in an XML file. You can view this XML file in the script's data/datasets folder. The data files are represented by an `<index_name>.xml` file, such as 20.xml. Since one file may contain several data tables, see the file **datasets.grd**, which maps the script index to the file index to determine which XML contains the data.

Recording WCF Duplex Communication

WCF (Windows Communication Foundation) is a programming model that unifies Web Services, .NET Remoting, Distributed Transactions, and Message Queues into a single Service-oriented programming model for distributed computing.

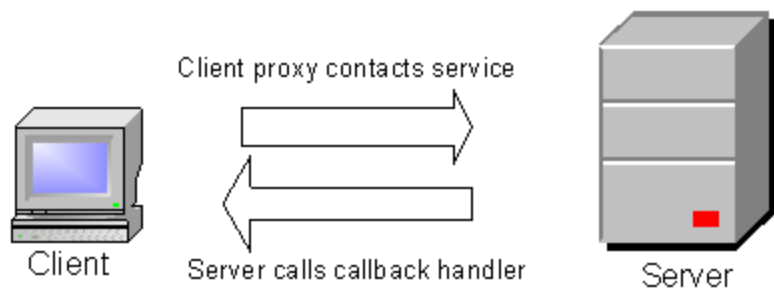
WCF creates a proxy object to provide data for the service. It also marshals the data returned by the service into the form expected by the caller.

In addition to general support for the WCF environment, VuGen provides specialized support for applications that use WCF's duplex communication. In duplex communication, the client proxy contacts the service, and the service invokes the callback handler on the client machine. The callback handler implements a callback interface defined by the server. The server does not have to respond in a synchronous manner—it independently determines when to respond and invoke the callback handler.

Communication Between Client and Server

The communication between the client and server is as follows:

- The server defines the service contract and an interface for the callback.
- The client implements the callback interface defined by the server.
- The server calls the callback handler in the client whenever needed.



When trying to record and replay duplex communication, you may encounter problems when the script calls the original callback methods. By default, the callback handlers are not included in the filter. You could customize the filter to include those callback handlers. However, the standard playback would be ineffective for a load test, since many of the callbacks are local operations such as GUI updates. For an effective load test you cannot replay the original callback method invoked by the server.

VuGen's solution is based on replacing the original callback handler with a dummy implementation. This implementation performs a typical set of actions that you can customize further for your application.

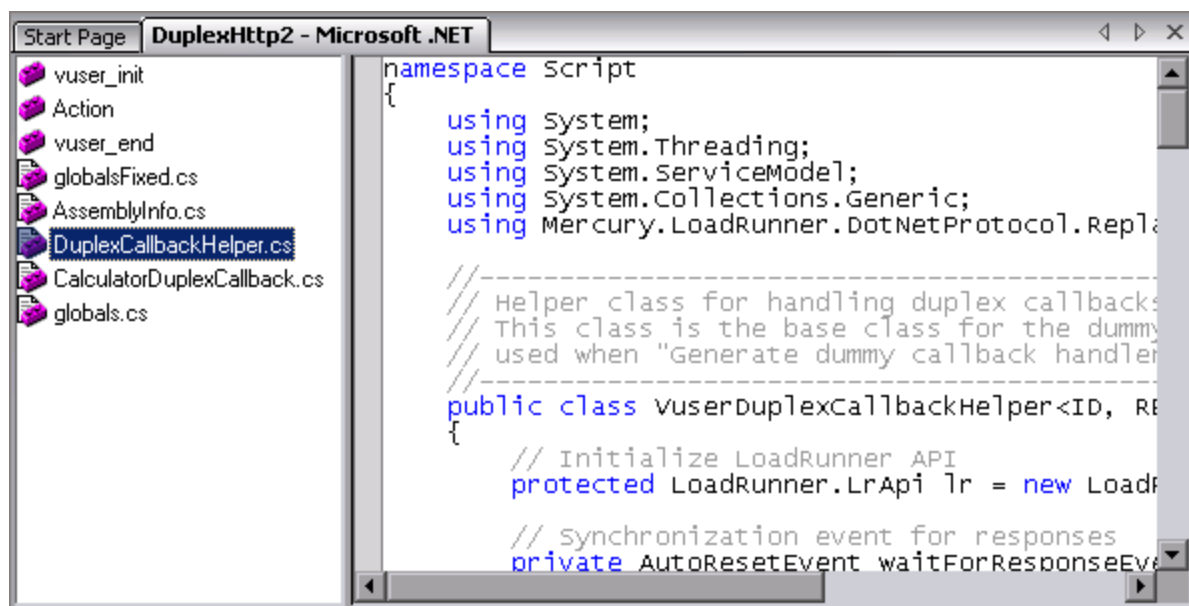
You instruct VuGen to replace the original callbacks by activating the **Generate Dummy Callback Handler** recording option. For more information, see ["Microsoft .NET > Recording - Recording Options" on page 182](#).

VuGen Implementation of a Duplex Callback

As part of the duplex communication solution, VuGen generates two support files:

- `DuplexCallbackHelper.<language>`
- `Callback_Name.<language>`

The following example shows the generated files for a Calculator application using duplex communication:



The Helper file serves as a general template for working with duplex callback handlers. It serves as a base class for the implementation of the callback.

The second file, **Callback_Name**, contains the implementation of the callback. The name of the callback implementation class is **Vuser<xxxx>** where *xxxx* is the name of the callback interface and it inherits from the **VuserDuplexCallbackHelper** class defined in the Helper file. VuGen creates separate implementation files for each interface.

This file performs two primary tasks:

- **Set Response.** It stores the data that came from the server in a map. It stores them with sequential IDs facilitating their retrieval. This method is called from the implementation of the callback interface. The following sample code demonstrates the dummy implementation of a callback method named **Result**. The method's arguments are stored in the map as an object array.

```
public virtual void Result(string operation, double result) {  
    // Add here your own callback implementation and set the response data  
    SetResponse(responseIndex++, new object[] {  
        operation,  
        result});  
}
```

- **Get Response.** Waits for the next response to arrive. The implementation of `GetNextResponse` retrieves the next response stored in the map using a sequential indexer, or waits until the next response arrives.

The script calls `GetNextResponse` at the point that the original callback handler was called during recording. At that point, the script prints a warning to wait for here for the next response, and indicating the original callback.

Replacement of the Callback in the Script

When you enable the **Dummy Callback** option (enabled by default), VuGen replaces the original duplex callback handlers with dummy implementations. The dummy implementation is called `Vuser <Callback Name>`. At the point of the original callback handler, the script prints a warning indicating that it was replaced.

Customizing the Dummy Implementation

You can modify the implementation file to reflect your environment. This section contains several suggested customizations.

Timeouts

The default timeout for which the callback waits for the next response is 60000 msec, or one minute. To use a specific timeout, replace the call to **GetNextResponse** with the overloading method which gets the timeout as an argument as shown below. This method is implemented in the callback implementation file `<Callback_Name>` listed in the left pane after the **DuplexCallbackHelper** file.



```
Example: // Get the next response.  
// This method waits until receiving the response from the server  
// or when the specified timeout is exceeded.  
  
public virtual object GetNextResponse(int millisecondsTimeout) {  
    return base.GetResponse(requestIndex++, millisecondsTimeout);  
}
```

To change the default threshold for all callbacks, modify the **DuplexCallbackHelper** file.



```
Example:  
// Default timeout threshold while waiting for response  
protected int millisecondsTimeoutThreshold = 60000;
```

Key Identifier

Many applications assign key identifiers to the data, which connects the request and response to one another. This allows you to retrieve the data from the map using its ID instead of the built-in incremental index. To use a key identifier instead of the index, modify the file `<Callback_Name>` replacing the first base template parameter, **named ID**, with the type of your key identifier. For example, if your key identifier is a string you may change the first template argument from **int** to **string**:

```
public class VuserXXX : VuserDuplexCallbackHelper<string, object>
```

In addition, you may remove the implementation of `GetNextResponse()` and replace it with calls to `GetResponse(ID)` defined in the base class.

Return Values

By default, since VuGen supports *OneWay* communication, the implementation callback does not return any value or update an output parameter when it is invoked.

```
public virtual void Result(string operation, double result) {  
    // Add your own callback implementation and set the response data  
}
```

If your application requires that the callback return a value, insert your implementation at that point.

Get Response Order

In VuGen's implementation, a blocking method waits for each response. This reflects the order of events as they occurred during recording—the server responded with data. You can modify this behavior to execute without waiting for a response or to implement the blocking only after the completion of the business process.

Find Port

The **FindPort** method in the Helper file is a useful utility that can be used in a variety of implementations. The Helper class uses this method to find unique ports for running multiple instances of the script. You can utilize this utility method for other custom implementations.

Recording Server Hosted By Client Applications

If the communication in your system is a server hosted by a client, VuGen's default solution for duplex communication will not be effective. In server hosted by client environments, it is not true duplex communication since the client opens the service and does not communicate through the Framework. For example, in queuing, the client sends a message to the service and opens a response queue to gather the responses.

To emulate a server hosted by a client, use the pattern depicted in the above solution—replace the original response queues with dummy callbacks and perform synchronization as required. For more information, contact HPE support.

Asynchronous Calls

When VuGen records asynchronous calls on remote objects, you can specify how the calls are handled in the "[Microsoft .NET > Recording - Recording Options](#)" on [page 182](#). These options are particularly relevant for .NET Remoting and WCF environments.

You can configure VuGen to one of the following options:

- **Call original callbacks by default.** Uses the recorded application's original callback when generating and replaying the script. If the callback method is explicitly excluded by a filter, the callback will be excluded even if you enable this option. If your callbacks perform actions that are not directly related to the business process, such as updating the GUI, disable this option.
- **Generate asynchronous callbacks.** This option defines how VuGen will handle callbacks when the original callbacks are not recorded. This is relevant when the above option, **Call original callbacks** is disabled or when the callbacks are explicitly excluded.

When you enable this option, it creates a dummy method which will be called during replay instead of the original callback. This dummy callback will be generated in the **callbacks.cs** section of the script.

When you disable this option, VuGen inserts a NULL value for the callback and records the events as they occur.

Note: VuGen supports the *Async* and *Await* modifiers, so if your AUT (application under test) uses these modifiers, they will be included in your script.

To display the callback method, `OnComplete1`, you click on the **callback.cs** file in the left pane.

Note: If you recorded a script with specific recording options, and you want to modify them, you do not need to re-record the script. Instead, use the **Record > Regenerate Script** option to recreate the script with the new settings.

Recording Dual HTTP Bindings

If your application employs dual HTTP Binding, since HTTP is inherently not a duplex protocol, the framework uses a standard port to receive response data being passed to the callback. When you attempt to run multiple instances of your application, you may be unable to do so using the same port number. VuGen provides you with an option of replacing the original client base address's port number with a unique port.

When you enable the **Generate Unique Client Base Address** recording option, VuGen checks the type of communication used by the application. If it detects dual HTTP communication, **WSDualHttpBinding**, it runs the **FindPort** utility (provided in LrReplayUtils) in the Helper file and finds unique ports for each instance of the callback.

This option is enabled by default. It is only relevant when you enable the above option, **Generate dummy callback handler**.

When you enable this option, VuGen generates the following code in the script:

```
#warning: Code Generation Warning
// Override the original client base address with a unique port number
DualProxyHelper.SetUniqueClientBaseAddress<XXXX>(YYYYY);
```

For more information, see ["Microsoft .NET > Recording - Recording Options" on page 182](#).

Connection Pooling

ADO.NET providers deploy a feature called **connection pooling** which can significantly influence load test accuracy. Whenever only one app domain is used for all Vusers, connection pooling is turned on—.NET Framework keeps the database connections open and tries to reuse them when a new connection is requested. Since many Vusers are executed in the context of a single application domain, they may interfere with one another. Their behavior will not be linear and that may decrease their accuracy.

In the .NET runtime settings, the AppDomain Per Vuser property enables execution of each Vuser in a separate app domain (true by default). This means that there is connection pooling in the scope of each Vuser, but the Vusers will not interfere with one another. This setting provides more accuracy, but lower scalability.

If you disable this option, you need to manually disable connection pooling for the database.

The following table describes how to manually disable connection pooling:

Provider	Option
----------	--------

.NET Framework Data Provider for SQL Server	"Pooling=false" or "Pooling=no"
.NET Framework Data Provider for Oracle	"Pooling=false" or "Pooling=no"
.NET Framework Data Provider for ODBC	Connection pooling is managed by an ODBC Driver Manager. To enable or disable connection pooling, use the ODBC Data Source Administrator (found in Control Panel or the Administrative Tools folder). The Connection Pooling tab allows you to specify connection pooling parameters for each of the installed ODBC drivers.
.NET Framework Data Provider for OLE DB	"OLE DB Services=-2"
Oracle Data Provider for .NET	"Pooling=false"
Adaptive Server Enterprise ADO.NET Data Provider	"Pooling=False"

Debugging .NET Vuser Scripts

You can compile a .NET Vuser script to check its syntax, without running the script. To compile the script directly from VuGen, press Shift+F5 or select **Vuser > Compile**. If VuGen detects a compilation error, it displays the error in the Output window. Double-click on the error to go to the problematic line in the script.

To run the script directly from VuGen, press F5 or select **Replay > Run**. Breakpoints and step-by-step replay are not supported in VuGen's editor window for .NET Vusers. To debug a script and run it with breakpoints or step-by-step, run it from within Visual Studio .NET as described below.

Viewing Scripts in Visual Studio

Visual Studio provides you with additional tools to view, edit, and debug your Vuser scripts. You can add breakpoints, view variable values, add assembly references, and edit the script using Visual Studio's IntelliSense. You can also run the script in a step-by-step mode for debugging.

When you save your script, VuGen creates a Visual Studio solution file, **Script.sln**, in your script's folder. You can open the solution file in Visual Studio .NET and view all of its components in the Solution Explorer.

To open the Vuser script in Visual Studio, select **Design > Open Script in Visual Studio** or click the **Visual Studio** button  on the VuGen toolbar.

Note: By default, the Vuser script opens in the latest version of Visual Studio that is installed on your computer. For example, if you have both Visual Studio 2013 and 2015 installed, the script will be converted and opened in Visual Studio 2015.

Double-click the appropriate section in the Solution Explorer, such as **vuser_init.cs**, to view the contents of the script.

Note: VuGen automatically loads all of the necessary references that were required during recording. You can add additional references for use during compilation and replay through the Solution Explorer. Select the **Reference** node and select **Add Reference** from the right-click menu.

Click on **globals.cs** or **globals.vb** in the Solution Explorer to view a list of the variables defined and used by your script.

.NET Filters Overview

Recording filters indicate which assemblies, interfaces, namespaces, classes, or methods to include or exclude during the recording and script generation.

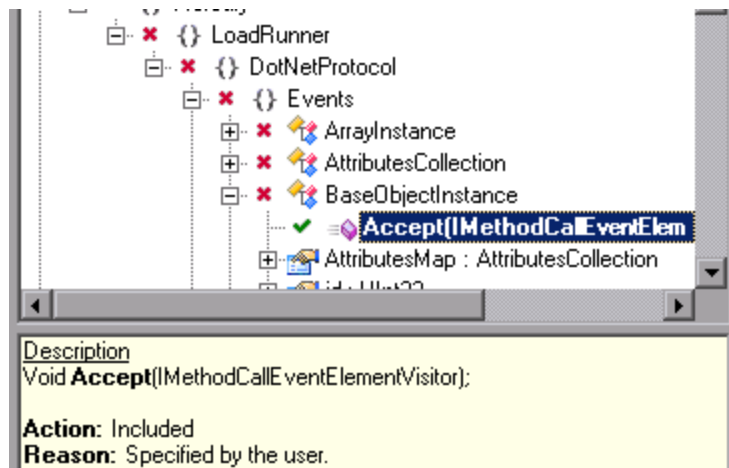
By default, VuGen provides built-in system filters for .NET Remoting, ADO.NET, Enterprise Services, and WCF (Windows Communication Foundation). These filters were designed to include the relevant interfaces for standard ADO.NET, Remoting, Enterprise Services, and WCF. VuGen also allows you to design custom filters.

Custom filters provide several benefits:

- **Remoting.** When working with .NET Remoting, it is important to include certain classes that allow you to record the arguments passed to the remote method.
- **Missing Objects.** If your recorded script did not record a specific object within your application, you can use a filter to include the missing interface, class or method.

- **Debugging.** If you receive an error, but you are unsure of its origin, you can use filters to exclude methods, classes, or interfaces in order to pin-point the problematic operation.
- **Maintainability.** You can record script in higher level, make script more easy to maintain and to correlate.

The **.NET Recording Filter pane** lets you manipulate existing custom filters. It displays the assemblies, namespaces, classes, methods, and properties in a color-coded tree hierarchy.



The bottom pane provides a description of the assembly, namespace, class, method, property, or event. It also indicates whether or not it is included or excluded and a reason for the inclusion or exclusion.

See also:

- [".NET Recording Filter Pane" on page 83](#)

Guidelines for Setting .NET Filters

When testing your .NET application, your goal is to determine how the server reacts to requests from the client. When load testing, you want to see how the server responds to a load of multiple users.

When recording a .NET application, your script may include calls to methods that do not affect the server, such as calls to a local utility or the GUI interface. These calls are usually not relevant to your testing goals, and it would be correct to filter them out.

The built-in filters, .NET Remoting, ADO.NET, Enterprise Services, and WCF, were designed to record only the server related traffic relevant to your testing goals. In some instances, however, you may need to customize filters to capture your .NET application's calls or exclude unnecessary calls. Using the [".NET Recording Filter Pane" on page 83](#), you can design custom filters to exclude the irrelevant calls and capture the server related calls.

Before creating a test, we recommend that you become familiar with your application and determine its primary classes and methods, so that you will know which ones to include in your recording.

If you are not familiar with your application's classes, you can use **Visual Studio** or a **Stack Trace** to help you determine which methods are being called by your application in order to include them in the filter.

VuGen allows you to record with a stack trace that logs all of the methods that were called by your application.

Once you determine the required methods and classes, you include them using the .NET Recording Filter pane. When preparing a script, you may need to customize the filter several times in order to achieve the optimal filter. An optimal filter records the relevant methods without introducing a large number of irrelevant calls to the script.



Tip: Strive to modify the filter so that your script will compile (Shift+F5) inside VuGen. Then customize the filter further to create a functional script that runs inside VuGen.

If you plan to add manual code to your script such as control flow or message statements, make sure to do so after you have a functional script that runs inside VuGen. The reason for this is that if you re-record a script or regenerate the script, you will lose all of the manual changes.

Determining which Elements to Include or Exclude

When designing a custom filter, we recommend that you start by choosing the appropriate built-in filter as a base filter. You can then customize the filter using one of the following approaches:

- **Top Down Approach.** An approach in which you include the relevant namespace and exclude specific classes that are not part of the client-server activity. This is recommended if you are familiar with your application and you can identify a well-defined assembly which implements all client-server activity without involving any GUI elements, such as MyDataAccessLayer.dll.
- **Bottom up Approach.** An approach in which you use the default filter and refine it by adding individual methods or classes. Use this approach if you cannot identify a well-defined layer or if you are not familiar with your application. Do not add all AUT assemblies and then try to remove extra component one by one.

The following guidelines indicate when to include or exclude elements:

- If, as a result of your including a class, your script has many unrelated method calls, try modifying the filter to exclude the irrelevant methods.
- If you identify a non-client/server call in your script, exclude its method in the filter.
- During recording, VuGen may detect an unknown input argument, for example, an argument whose construction it had never encountered before. If this argument supports serialization, VuGen **serializes** it by saving it to a file in a special format. During replay, VuGen reconstructs the argument by **deserializing** it.
- VuGen serializes objects passed as arguments that were not included by the filter. We recommend that you include this object in the filter in order to track its construction and activity instead of using it in its serialized form. You can identify serialized objects in the script by searching for calls to the **LrReplayUtils.GetSerializedObject** method or, in WCF environments, **LrReplayUtils.GetSerializedDataContract**. VuGen stores serialized objects in the script's **\data\SerializedObjects** folder as XML files with indexes: **Serialization_1.xml**, **Serialization_2.xml** and so forth.
- When no rules are specified for a method, it is excluded by default. However, when the remoting

environment is enabled, all remote calls are included by default, even if they are not explicitly included. To change the default behavior, you can add a custom rule to exclude specific calls which are targeted to the remote server.

- Arguments passed in remoting calls whose types are not included by the filter, are handled by the serialization mechanism. To prevent the arguments from being serialized, you can explicitly include such types in order to record the construction and the activity of the arguments.
- Exclude all activity which involves GUI elements.
- Add assemblies for utilities that may be required for the script to be compiled.

For information on how to include and exclude elements, see [".NET Recording Filter Pane" on page 83](#).



Tip: You can include or exclude a method directly from the VuGen editor. Select the method and choose **Open current method in Filter Pad** from the right-click menu. Once the method is selected in the filter tree, choose **Include** or **Exclude** from the right-click menu.

Defining an Effective Filter

When preparing a script, you may need to customize the filter several times in order to achieve the optimal filter. An optimal filter records the relevant methods without introducing a large number of irrelevant calls to the script.

To define an effective filter:

1. Create a new filter based on one of the built-in filters. If you know that the AUT (Application Under Test) does not use ADO.NET, Remoting, WCF, or Enterprise Services, clear that option since unnecessary filters may slow down the recording.
2. Set the **Stack Trace** option to true for both recording and code generation. Open the Recording Options (ctrl+f7) and select the **Recording** node. Enable **Debug Options: Stack Trace** and **Code Generation: Show Stack Trace**.
3. Record your application. Click **Start Record** (ctrl + r) to begin and **Stop** (ctrl + f5) to end.
4. View the script's steps. If you can determine the business logic from the steps and apply correlation, you may not need to create custom filters. If, however, the script is very long or hard to maintain or correlate, you should customize the script's filter.
5. Try to identify the high-level method in the call that captures or wraps one or more client server calls. You can do this by opening the AUT source files (if they are available) in Visual Studio or by viewing a Stack Trace of the script.
6. Set the filter to include the relevant methods—you may need to add their assembly beforehand. For tips about including and excluding elements in the filter, see [".NET Filters Overview" on page 583](#).
7. Record the application again. You should always re-record the application after modifying the filter.
8. Repeat steps 4 through 7 until you get a simple script which can be maintained and correlated.
9. After creating an optimal script, turn off the **Stack Trace** options and regenerate the script. Open

the Recording Options (ctrl+f7) and select the **Recording** node. Disable **Debug Options: Stack Trace** and **Code Generation: Show Stack Trace**. This will improve the performance of subsequent recordings.

10. Correlate the script. For your test to run properly, you may need to insert a correlation to capture a value and use it at a later point in the script. For more information, see ["Microsoft .NET Correlation" on page 263](#).

.NET Filters - Advanced

The filter tree hierarchy only displays public classes and methods. It does not show non-public classes or delegates.

You can add classes or methods that are not public by manually entering them in the filter's definition file.

The filter definition files, **<filter_name>.xml** reside in the dat\DotnetFilters folder of your installation. The available Action properties for each element are: **Include**, **Exclude**, or **Totally Exclude**. For more information, see [".NET Recording Filter Pane" on page 83](#).

By default, when you exclude a **class**, the filter mechanism applies **Exclude**, excluding the class, but including activity generated by the excluded class. When you exclude a **method**, however, it applies **Totally Exclude**, excluding all referenced methods.

A screenshot of a text editor window displaying an XML configuration file for .NET filters. The XML is structured with nested elements for namespaces, classes, and methods, each with an 'Action' attribute. The root element is '</Configuration>'. Inside, there's '</Filter>', followed by several '</Namespace>' and '</Class>' closing tags. The main configuration starts with '<Namespace Name="Microsoft" Action="Exclude">'. Inside this, there's '<Namespace Name="MediaCenter" Action="Default">', which contains '<Namespace Name="TV" Action="Default">'. This in turn contains '<Namespace Name="Tuning" Action="Default">'. Inside 'Tuning', there are two class definitions: '<Class Name="AnalogAudioComponentType" Action="Default">' with a method '<Method Name=".ctor()" Action="TotallyExclude"/>', and '<Class Name="AnalogLocator" Action="Exclude">' with a method '<Method Name="get_VideoStandard()" Action="TotallyExclude"/>'. The XML is color-coded with red for tags, blue for attributes, and black for text. The editor has a standard Windows-style interface with a title bar, menu bar, and toolbar.

For example, suppose Function A calls function B. If Function A is **Excluded**, then when the service calls Function A, the script will include a call to Function B. However, if function A is **Totally Excluded**, the script will not include a call to Function B. Function B would only be recorded if called directly—not through Function A.

VuGen saves a backup copy of the filter as it was configured during the recording, **RecordingFilterFile.xml**, in the script's **data** folder. This is useful if you made changes to the filter since your last recording and you need to reconstruct the environment.

Reference List Dialog Box [.NET Protocol]

This dialog box allows you to manage the list of all of the referenced DLLs for your script.

To access	Click within the VuGen editor and select References from the right-click menu.
------------------	---

User interface elements are described below:

Column	Description
Active	Entries in the list are global for all .NET scripts. This check box allows you to indicate whether this file should be active for the current script.
Shared	Indicates whether or not the DLL is shared.
Name	The name of the DLL, usually containing its details, such as version number and token information (read-only). <div> <p>Note: If you add a reference to a DLL for a .NET filter, VuGen adds it to the script's reference list only if the script accesses the DLL's methods during recording.</p> </div>
Copy Local	Indicates whether or not to copy the DLL locally. Clear this check box if the DLL loads the library by itself, or if it is available system wide, such as a DLL registered in GAC.
Hint Path	The HintPath of the project file, if specified (read-only).
Is Specific Version	Indicates whether or not the DLL is for a specific version.


Note: If you make any changes, such as selecting **Copy Local**, removing a row, or disabling **Active** for a specific reference, they will not be effective until you regenerate the script (**Record > Regenerate Script**).

See also:

- [.NET Recording Filter Pane](#)

Add Reference Dialog Box [.NET Protocol]

This dialog box enables you to add references to your .NET script or to the .NET filters.

To access	<p>To add a reference to your script:</p> <ol style="list-style-type: none"> 1. In a .NET script editor, select References from the right-click menu. 2. In the References List window, click Add Reference. <p>To add a reference to a .NET filter:</p> <ol style="list-style-type: none"> 1. Select View > .NET Recording Filter or click . 2. Select the Custom filters option in the left pane. 3. Click New to create a new filter. Select an option and click OK. 4. In the .NET Recording Filter pane, select a filter element. 5. Click Add Reference.
------------------	--

User interface elements are described below (unlabeled elements are shown in angle brackets):


UI Element	Description
<Component List>	<p>A list of .NET Framework components or assemblies in the Public Assemblies folder.</p> <ul style="list-style-type: none"> • To add one of the listed items, select it and click Select. You can select multiple components using Ctrl-click. The bottom pane shows the selected references. • To add an assembly that is not in the list, click Browse and locate the reference on your file system or network.
<Selected Component List>	<p>The list of selected components. The Type column indicates .NET for a component from the Public Assemblies folder and File for a component that was added by selecting Browse.</p> <ul style="list-style-type: none"> • To clear an item from the list, select it in the bottom pane and click Remove.

See also:

- [.NET Recording Filter Pane](#)

Create a New Filter Dialog Box [.NET Protocol]

This dialog box enables you to create a new filter for .NET Vuser scripts.

To access	Perform the following: <ol style="list-style-type: none">1. Select View > .Net Recording Filter or click the  toolbar button.2. Select the Custom filter option in the left pane.3. Click New.
------------------	---

User interface elements are described below:

UI Element	Description
Start with an empty filter	Create a new filter that is not based on a pre-existing filter.
Based on an environment filter	Create a new filter based on an environment filter. Use the check boxes next to the environment filters to indicate which environment filters to base the filter on.
Based on a custom filter	Create a new filter based on a custom filter. Use the drop-down menu to select the custom filter.

 **See also:**

- [.NET Recording Filter Pane](#)

Configure Application Security and Permissions

A Security Exception that occurs while recording an application is usually due to a lack of permissions—the recording machine does not have sufficient permissions to record the application. This is common where your application is not local, but on the Intranet or network.

To solve this problem, you need to allow the recording machine to access the application and the script with Full Trust.

One solution is to copy the application and save your script locally, since by default, users have Full Trust permissions to all local applications and folders.

An additional solution is to create new code groups that gives Full Trust to each application folder, and the script folder.

Grant Full Trust permissions to a Specific Folder (Visual Studio NOT installed)

1. From the command prompt, run the caspol.exe application.
2. Set the desired permission.

Grant Full Trust Permissions to a Specific Folder (Visual Studio installed)

1. Open the .NET Configuration settings. Select **Start > All Programs > Administrative Tools > Microsoft .NET Framework 2.0 Configuration**. The .NET Configuration window opens.

2. Expand the **Runtime Security Policy** node to show the Code Groups of the machine.
3. Select the **All_Code** node.
4. Select **Action > New**. The Create New Code Group dialog box opens.
5. Enter a name for a new Code Group for your application or script. Click **Next**.
6. Select the **URL** condition type. In the URL box, specify the full path of the application or script in the format `file:///...` and click **Next**.
7. Select the **FullTrust** permission set. Click **Next**.
8. Click **Finish** in the Completing the Wizard dialog box. The configuration tool adds your Code Group to the list of existing groups.
9. Repeat the above procedure for all .NET applications that you plan to record.
10. Repeat the above procedure for the Vuser script folder.

Note: Make sure that the script folder has **FullTrust** permissions on all Load Generator machines that are participating in the test (LoadRunner only).

Troubleshooting and Limitations - .NET

This section describes troubleshooting and limitations for .NET Vuser scripts.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

Replay Limitations

- Exceptions in .NET scripts generated by vts functions, may cause the replay to abort, even when the **Continue on error** Runtime setting is enabled.
Workaround: Use `try` and `catch` statements for error handling of the vts functions.
- .NET Scripts which contain UI objects can behave unexpectedly during replay in VuGen, the Controller, or on a load generator machine.

Recording Limitations

The following limitations apply to the VuGen recording of a Microsoft .NET application:

- .NET Vuser scripts support only single-protocol recording in VuGen.
- Direct access to public fields is not supported—the AUT must access fields through methods or properties.
- VuGen does not record static fields in the applications—it only records methods within classes.
- Multi-threaded support is dependent on the client application. If the recorded application supports multi-threading, then the Vuser script will also support multi-threading.

- In certain cases, you may be unable to run multiple iterations without modifying the script. Objects that are already initialized from a previous iteration, cannot be reinitialized. Therefore, to run multiple iterations, make sure to close all of the open connections or remoting channels at the end of each iteration.
- Recording is not supported for Enterprise Services communication based on MSMQ and Enterprise Services hosted in IIS.
- The .NET version should be the same on the record and replay machines.
- VuGen partially supports the recording of WCF services hosted by the client application.
- Recording is not supported for Remoting calls using a custom proxy.
- Recording is not supported for **ExtendedProperties** property of ADO.NET objects, when using the default ADO.NET filter.
- Applications created with .NET Framework 1.1 which are not compatible with Framework 2.0, cannot be recorded. To check if your Framework 1.1 application is compatible, add the following XML tags to your application's .config file:

```
<configuration>
  <startup>
    <supportedRuntime version="v2.0.50727"/>
  </startup>
</configuration>
```

Invoke the application (without VuGen) and test its functionality. If the application works properly, VuGen can record it. Remove the above tags before recording the AUT with VuGen. For more information regarding this solution, see the MSDN Knowledge Base.

- Applications that use the .NET Remoting Framework and are executed in CLR 2 (.NET frameworks 2/3/3.5), might crash during recording. During a crash you will receive a message containing the strings Version=4.x.x.x, and "is not registered for activation".
Potential Workaround: In the Microsoft .NET: Recording user interface under Support for previous.NET version, select **Emulate previous .NET versions in transport level**, and then record again.
- When the application under test retrieves a server-activated object by calling new RemoteObject(), VuGen generates a RemotingServices.Connect function.
- Applications using multiple processes or multiple application domains are only partially supported.
- Shared DLLs must be specified in the Recording Options only. Changes made in the runtime settings to the list of shared DLLs have no effect.
- When McAfee antivirus is active, it may issue the following message when recording a .NET script: "The solution has been changed externally".
Workaround: Add the **VuGen.exe** process to the Low-Risk processes in the McAfee antivirus **On-Access Scan Properties**.
- VuGen does not record a method that gets a private class as an argument. For example:

```
public class A
```

```
{  
    private class P  
    {  
        ...  
    }  
    private void MethodUsingPrivateClass(P p)  
    {  
        ...  
    }  
}
```

- References added to a .NET script during recording, by means of the .NET Recording Filter, are not removed from the script by removing them in the References List dialog box.

Script Editing Limitations

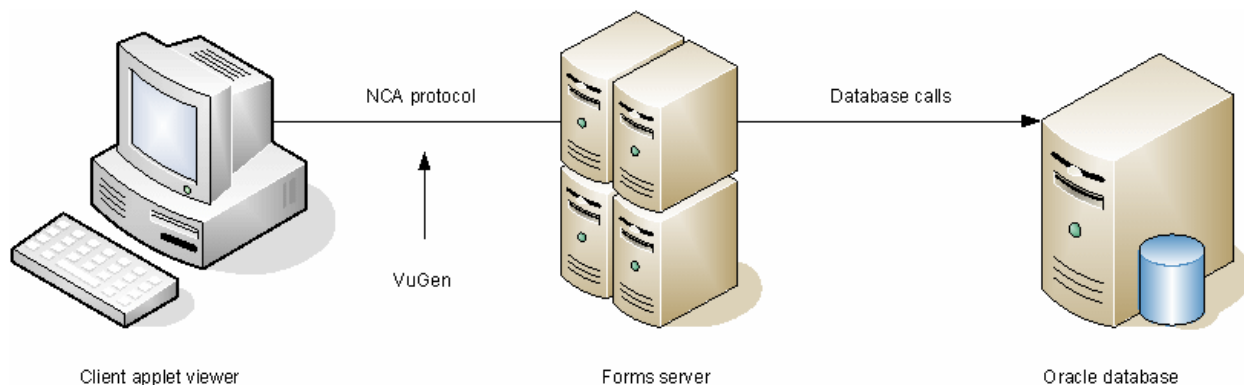
Disable FIPS if you want to open or debug a .NET Vuser script in Visual Studio 2015.

Oracle NCA Protocol

Oracle NCA Protocol Overview

Oracle NCA is a protocol that handles communication with the Oracle Forms server. Using your browser, you launch the database client, an applet viewer. You perform actions on the NCA database through its applet viewer. This eliminates the need for client software and allows you to perform database actions from all platforms that support the applet viewer.

The NCA environment is a three-tier environment. The user first sends an http call from his browser to a Web server. This call accesses the startup HTML page which invokes the Oracle Applications applet. The client (applet viewer) communicates through the proprietary NCA protocol with the application server (Oracle Forms server) which then submits information to the database server.



VuGen records and replays the NCA communication between the client and the Forms server (application server).

The Oracle NCA protocol is commonly used as a multi-protocol in combination with Web - HTTP/HTML. This is the recommended way to record with Oracle NCA. If you are using Oracle NCA as a single protocol, web events are recorded but steps are not generated (or replayed) by default.

If you initially created a single protocol script for Oracle NCA, and at a later stage you require the Web functions for testing, you can regenerate your script in VuGen to add the Web functions, without having to re-record the session. You indicate this from the Protocols node in the Recording Options.

Oracle NCA Protocol Example Scripts

In the following example, the user selected an item from a list (**nca_list_activate_item**), pressed a button (**nca_button_press**), retrieved a list value (**nca_lov_retrieve_items**), and performed a click in an edit field (**nca_edit_click**). The logical names of the objects are the parameters of these functions.

```
nca_lov_select_item("Responsibilities","General Ledger, Vision Operations");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","+ Journals");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0"," Enter");
nca_button_press("GLXJEENT.TOOLBAR.LIST.0");
nca_lov_find_value("Batches","");
nca_lov_retrieve_items("Batches",1,9);
nca_lov_select_item("Batches","AR 1020 Receivables 2537: A 1020");
nca_edit_click("GLXJEENT.FOLDER_QF.BATCH_NAME.0");
```

In certain tests, such as those performed on Oracle Configurator applications, information returned by one function is required throughout the session. VuGen automatically saves the dynamic information to a parameter, by inserting a **web_reg_save_param** function into the script. In the following example, the connection information is saved to a parameter called NCAJServSessionID.

```
web_reg_save_param ("NCAJServSessionId", "LB=\r\n\r\n", "RB=\r",
    LAST);
web_url("f60servlet",
    "URL=http://usscifforms05.sfb.na/servlet/f60servlet\?config=mult",
    LAST);
```

In the above example, the right boundary is \r. The actual right boundary may differ between systems.

Note: We recommend that the user not modify the **web_reg_save_param** parameters if they were generated automatically. Alternatively, you can manually add a new **web_reg_save_param** function or add a new correlation rule.

Oracle NCA Record and Replay Tips

When recording an Oracle NCA Vuser script, follow these guidelines:

- We recommend installing Jinitiator before recording a script.
- Close all browsers before you begin recording.
- Record the login procedure in the **vuser_init** section. Record a typical business process in the Actions section. When you run the script, you can then specify multiple iterations for a specific business process. For more information, see ["Solution Explorer Pane" on page 46](#).
- VuGen supports the recording of Oracle Forms applications using the Forms Listener Servlet in multi-protocol mode. The application server uses the **Forms Listener Servlet** to create a runtime process for each client. The runtime process, **Forms Server Runtime**, maintains a persistent connection with the client and sends information to and from the server.
- To support Forms 4.5 in replay, modify the **mdrv_oracle_nca.dat** file in the **dat > mdrv** folder to match the following example:

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp110.dll
WIN95_EXT_LIBS=ncarp110.dll
LINUX_EXT_LIBS=liboranca.so
SOLARIS_EXT_LIBS=liboranca.so
HPUX_EXT_LIBS=liboranca.sl
AIX_EXT_LIBS=liboranca.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api
```

To restore Forms support for versions later than 4.5, restore the original values.

Pragma Mode

The client side of the Oracle NCA Vuser can be configured to send an additional header to the server named **Pragma**. The header is a counter that behaves in the following way: the initial message of the NCA handshake has a value of 1.

The messages that follow the handshake are counted, beginning with 3. The counter is incremented by 1 for each message sent by the client.

If the message received from the server is the type `plain\text` and the body of the message begins with `ifError:#!/#00`, the client sends a 0 byte message to the server and the Pragma value changes its sign to a minus. This sign changes back after the client succeeds in receiving the information from the server.

Recording of the Pragma header is only supported in the multi-protocol mode (Oracle NCA and Web). You can identify the Pragma mode within the script's default.cfg file. When operating in Pragma mode, the `UseServletMode` is set to 2.

```
[HttpConnectMode]
UseHttpConnectMode=1
RelativeURL=<NCAJServSessionId>
```

```
UseServletMode=2
```

For information on the Pragma related settings, see the **Oracle NCA > Client Emulation** view in the runtime settings.

To identify the Pragma mode, you can perform a WinSock level recording and check the buffer contents. In the first example, the buffer contains the Pragma values as a counter:

```
send buf108
  "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
  "vlet=gk5q79uqy1 HTTP/1.1\r\n"
  "Pragma: 1\r\n"
  ...
send buf110
  "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
  "vlet=gk5q79uqy1 HTTP/1.1\r\n"
  "Pragma: 3\r\n"
  ...
```

In the following example, the buffer contains the Pragma values as an error indicator:

```
recv buf129 281
  "HTTP/1.1 200 OK\r\n"
  "Date: Tue, 21 May 2002 00:03:48 GMT\r\n"
  "Server: Oracle HTTP Server Powered by Apache/1.3.19 (Unix) mod_fastcgi/2.2"
  ".10 mod_perl/1.25 mod_oprocmgr/1.0\r\n"
  "Content-Length: 13\r\n"
  "Content-Type: text/plain\r\n"
  "\r\n"
  "ifError:8/100"
send buf130
  "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
  "vlet=gk5q79uqy1 HTTP/1.1\r\n"
  "Pragma: -12\r\n"
  ...
```

Enable the Recording of Objects by Name

When recording an Oracle NCA script, you must record the session using object names instead of the standard object ID. If the script is recorded using the object ID, replay may fail because the ID is generated dynamically by the server and may differ between iterations. You can verify that your script is being recorded with object names by examining the **nca_connect_server** function.

```
nca_connect_server("199.35.107.119", "9002"/*version=11i*/, "module=/d1/oracle
  /visappl/fnd/11.5.0/forms/US/FNDSCSGN userid=APPLSYSPUB/PUB@VIS
  fndnam=apps record=names ");
```

If the **record=names** argument does not appear in the **nca_connect_server** function, you may be recording object IDs. You can instruct VuGen to record object names by modifying one of the following:

Startup HTML file

If you have access to the startup HTML file, instruct VuGen to record object names instead of its object ID by setting the **record=names** flag in the startup file—that is, the file loaded when you start the Oracle NCA application:

1. In the startup file that is called when the applet viewer begins, locate the following line:

```
<PARAM name="serverArgs ... fndnam=APPS">
```

2. Add the Oracle key, **record=names**:

```
<PARAM name="serverArgs ... fndnam=APPS record=names">
```

Forms Configuration file

If the application has a startup HTML file that references a Forms Web CGI configuration file, **formsweb.cfg** (a common reference), you may encounter problems if you add **record=names** to the Startup file. In this case, modify the configuration file. To enable the recording of object names using the configuration file:

1. Open the Forms Web CGI configuration file.
2. Define a new parameter in this file (see sample Web CGI configuration file below for this change).

```
serverApp=forecast  
serverPort=9001  
serverHost=easgdev1.dats.ml.com  
connectMode=socket  
archive=f60web.jar  
archive_ie=f60all.cab  
xrecord=names
```

3. Open the startup HTML file and locate `PARAM NAME="serverArgs"`.
4. Add the variable name as an argument to the ServerArgs parameter, for example, **record=%xrecord%**:

```
<PARAM NAME="serverArgs" VALUE="module=%form% userid=%userid% %otherParams%  
record=%xrecord%">
```

5. Alternatively, edit the **basejini.htm** file in the Oracle Forms installation folder. This file is the default HTML file for running a form on the web using JInitiator-style tags to include the Forms applet. In the `basejini.htm` file add the following line to the parameter definitions:

```
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
```

In the <EMBED> tag, add the following line:

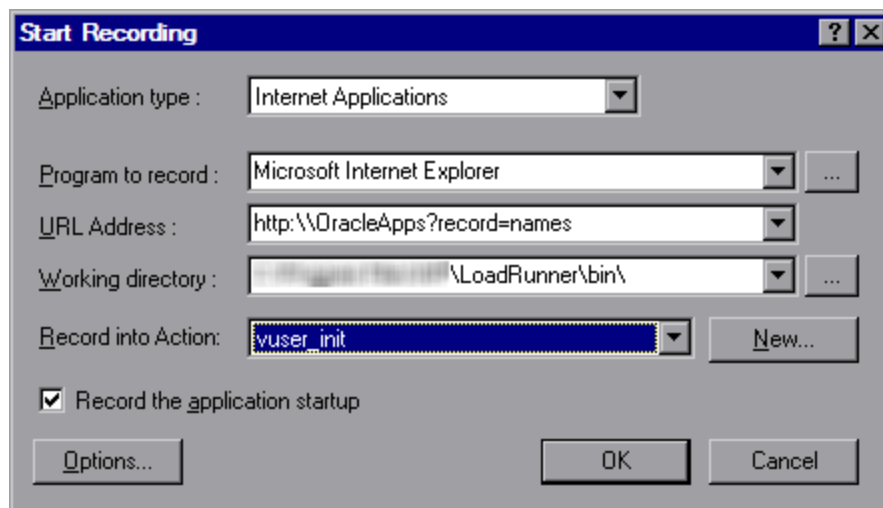
```
serverApp="%serverApp%"  
logo="%logo%"  
imageBase="%imageBase%"  
formsMessageListener="%formsMessageListener%"  
recordFileName="%recordFileName%"
```

The drawback in editing this file instead of the servlet configuration file **formsweb.cfg**, is that this file is replaced when you reinstall Oracle Forms. To avoid this, you can create a copy of the **basejini.htm** file and store it at another location. In the servlet configuration file, edit the **baseHTMLInitiator** parameter to point to the new file.

URL to record

If you do not have access to the startup HTML file, you can still have Oracle NCA record object names instead of its object ID by modifying the URL to record. The following solution only works if the startup HTML file does not reference another file while loading.

For this solution, you add **"?record=names"** after the URL in the Start Recording dialog box, after the URL name to record.



Alternatively, if you are using an application developed on Oracle Forms 10G or later, add **&otherparams=record=names&** to the URL.

Launch Oracle Applications via the Personal Home Page

When launching Oracle Forms applications (versions 6i and higher) by logging in through the **Personal Home Page**, you must set several system profile options at the user level. It is desirable to pass such

variables at the user level, and not at the site level, where it will affect all users. The following steps describe how to configure the "ICX: Forms Launcher" profile.

1. Sign on to the application and select the "System Administrator" responsibility.
2. Select **Profile/System** from the Navigator menu.
3. Within the **Find System Profile Values** form:
 - a. Select the **Display > Site** option
 - b. **Users** = <your user login> for example, operations, mfg, and so on)
 - c. **Enter Profile** =%ICX%Launch%
 - d. Click **Find**.
4. Update the User value to the **ICX:Forms Launcher** profile:
 - If no parameter has been passed to the URL, append the following string to the end of the URL of the user value: ?play==;record=names
 - If a parameter has been passed to the URL, append the following string to the end of the URL of the user value: =;play==;record=names
5. Save the transaction.
6. Log out of the Oracle Forms session.
7. Log out of the Personal Home Page session.
8. Sign on again via the **Personal Home Page** using your username.

If you were unable to update the ICX: Forms Launcher profile option at the user level, open the **Application Developer** responsibility and select the **Updatable** option for the ICX_FORMS_LAUNCHER profile.

The first parameter passed to the URL must begin with a question mark (?). You pass all subsequent parameters with an ampersand (&). In most cases, the URL already contains parameters, which you can identify by searching for a question mark.

Oracle - Troubleshooting and Limitations

This section describes troubleshooting and limitations for Oracle NCA and Oracle-Web protocol scripts.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

General Limitations

- The **Remote Application via LoadRunner Proxy > Display recording toolbar on client machine** option, is not supported for the Oracle NCA or Oracle-Web protocols. For details, see "[Start Recording Dialog Box](#)" on page 221.
- Recording by proxy (local or remote) works only for servers which use HTTP/S communication. It does not work for servers that use only Socket communication.

Testing Secure Oracle NCA Applications

- In the **Mapping and Filtering** node of the Recording Options dialog box, delete any existing Port mapping entries for port 443 and create a new Port mapping entry for the Oracle server name:

Service ID: HTTP

Target Server: Oracle Forms Server IP address or long host name

Target Port: 443

Connection Type: SSL

SSL Version: Active SSL version. If in doubt, select SSL 2/3.

Note that the **Service ID** is **HTTP** and not **NCA**.

For more information, see ["Network > Mapping and Filtering Recording Options" on page 185](#).

- If you encounter problems when replaying an NCA HTTPS script during the **nca_connect_server** command, insert the following function at the beginning of the script.

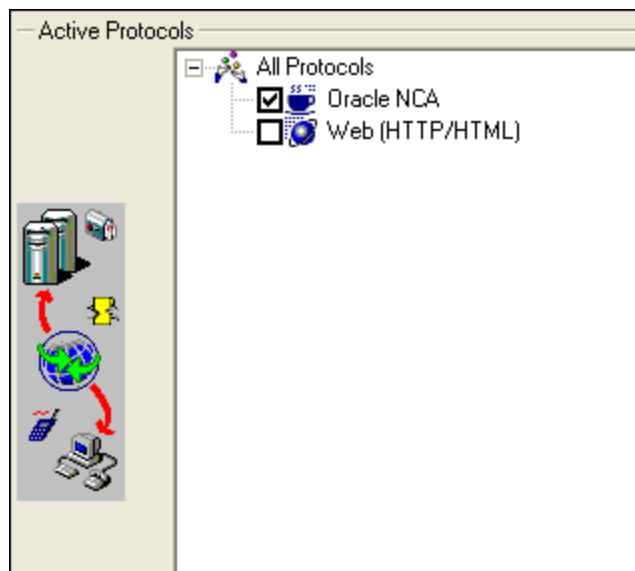
```
web_set_sockets_option("SSL_VERSION","3");
```

Testing Servlets and other Oracle NCA Applications

Certain NCA sessions use servlets:

- the Forms Listener servlet
- applications or modules that use both NCA and HTTP communications, such as the Oracle Configurator
- the initializing of the NCA application (downloading the applet, jar, and gif files)

When recording servlets, you must record both Oracle NCA and Web functions. You can do this by using the Oracle Apps Ili protocol or creating an Oracle NCA multi-protocol script. Alternatively, if you created a single protocol script for Oracle NCA, open the **General > Protocols** node in the Recording Options, and enable the Web protocol. Then you can begin recording.



If you are unsure whether your application uses servlets, you can check the **default.cfg** file in the script folder after recording a script. Locate the entry "**UseServletMode=**"

If the value is 1 or 2, then servlets are being used and you must enable HTTP recording in addition to Oracle NCA.

If you already recorded a script, you can regenerate the code automatically to include the Web functions without having to re-record. Select **Record > Regenerate Script**, and select the Web protocol in the Protocols section.

Determining the Recording Mode

When recording Oracle NCA scripts: VuGen automatically determines the correct connection mode: HTTP or Socket mode. Generally, you are not required to modify any of the recording settings as VuGen auto-detects the system configuration (unless you are working with Forms Server 4.5). In systems where the standard port mapping are reserved by other applications, you may need to modify the Port Mapping settings, depending on the recording mode.

You can determine the recording mode in one of the following ways:

- When using the NCA application, open the Java Console.

```
proxyHost=null
proxyPort=0
connectMode=HTTP
Forms Applet version is: 60812
```

The **connectMode** entry indicates **HTTP**, **HTTPS**, or **socket**.

- After recording an NCA session, open the **default.cfg** file in the Vuser folder and check the value of the **UseHttpConnectMode** entry.

```
[HttpConnectMode]  
UseHttpConnectMode= 2  
// 0 = socket 1 = http 2 = https
```

When defining a new port mapping in the Server Entry dialog box, use a **Service ID** of HTTP for HTTP or HTTPS modes. For Socket mode, use a **Service ID** of NCA.

For more information about Port Mapping settings, see ["Network > Mapping and Filtering Recording Options" on page 185](#).

Recording Trace Information for Oracle DB

To debug your script, you can use the Oracle DB breakdown graphs. To gather data for this graph, you turn on the trace mechanism before running the script.

To manually turn on the tracing mechanism, use the **nca_set_custom_dbtrace** function. For more information, see the Function Reference (**Help > Function Reference**).

RDP Protocol

RDP Protocol - Overview

The Microsoft RDP (Remote Desktop Protocol) enables one computer [the client] to connect to another computer [the server] over a network connection. For example, you can use RDP to connect to a central, powerful server for working on specific business applications or graphic terminals. This provides you with the same look and feel as if you are working on a standalone PC. The client computer employs RDP client software for this purpose, while the other computer must run RDP server software. The client software is referred to as **Remote Desktop Connection**. The server software is referred to as **Remote Desktop Services**.

For details on the versions of RDP that are supported by VuGen, see the [System Requirements](#).

Note: RDP versions 5.1 and later have an **Experience** tab that allows you to set various options. This tab is not supported by VuGen recording. All options are set to the ON position.

RDP Recording Tips

Note: This topic applies to RDP Vuser scripts only.

When recording an RDP Vuser script, follow these guidelines in order to create an effective script.

Single vs. Multi-Protocol Scripts

When creating a new script, you may create a single protocol or multi-protocol script. For example, to record both RDP traffic and Web responses, create a multi-protocol script for RDP and Web to enable the recording of both protocols.

Record into Appropriate Sections

Record the connection process into the **vuser_init** section, and the closing process into the **vuser_end** section. This will prevent you from performing iterations on the connecting and disconnecting processes. For more information about recording into sections, see ["Vuser Script Sections" on page 132](#).

FIPS

Beginning with LoadRunner version 12.50, recording is supported for RDP scripts when FIPS enforcement is enabled on the machine.

Run a Clean Session

When recording a session, make sure to perform the complete business process, starting with the connection and ending with the cleanup. End your session at a point from where you could start the entire process from the beginning. Do not leave any client or application windows open.

You should also configure your terminal server to end disconnected sessions. Select **Administrative Tools > Terminal Services Configuration > Connection Properties > Sessions > Override User Settings** and set the server to end disconnected sessions.

Explicit Mouse Clicks

When opening expanded menu options, click explicitly on each option—do not depend on the expanding menu. For example, when choosing **Start > All Programs > Microsoft Word**, be sure to click on the word **Programs**.

Using Windows Logo key combinations

Note: This tip applies only to Windows 8 installations on remote computers.

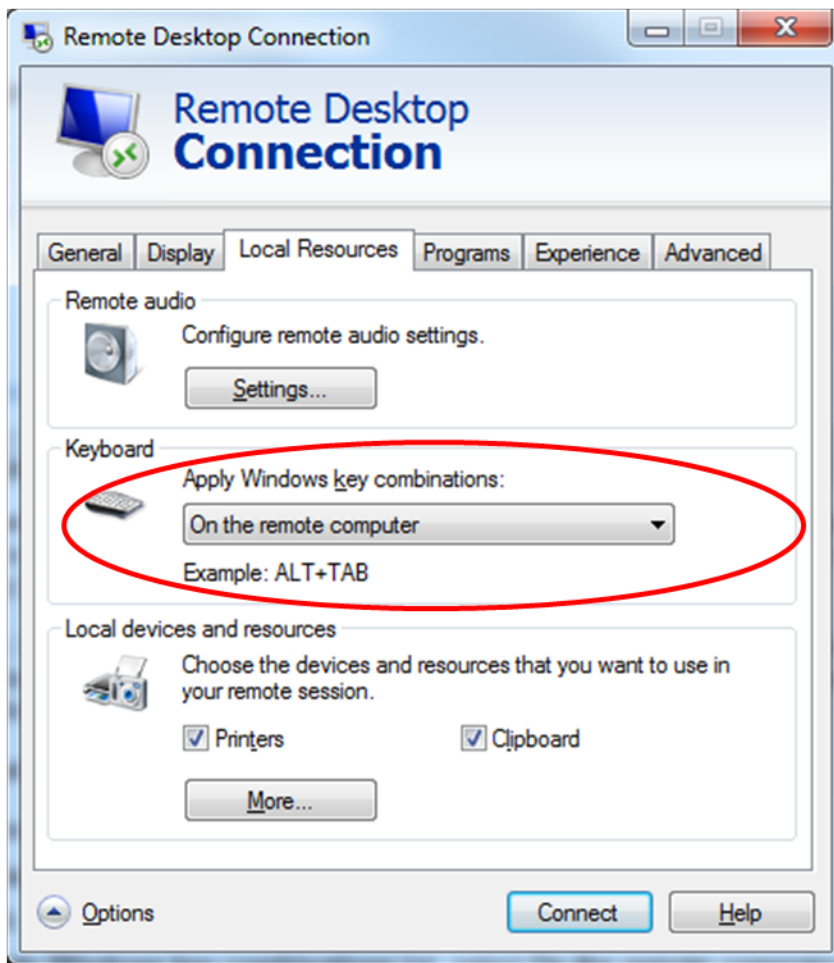
Because support for mouse movement in RDP Vusers can cause performance issues, by default, mouse movement support is disabled. Therefore, when you record an RDP Vuser script, it is recommended that you use the **Windows Logo** key combinations to display the Start screen [Windows Logo key], to show the Desktop [Windows logo key + D], and to open the charms bar [Windows Logo key + C].

When you run an RDP Vuser script, Windows key combinations can be applied on either the host computer or on the remote computer. To ensure that Windows key combinations are applied on the remote computer, when you record the connection to the remote computer, you must specify that Windows key combinations are applied on the remote computer.

How to apply Windows key combinations on the remote computer:

1. Open the Remote Desktop Connection dialog box.
2. Click **Options** to expand the dialog box.
3. Click the **Local Resources** tab.

4. Under **Keyboard**, from the **Apply Windows key combinations** list, select **On the remote computer**.



Synchronizing using Windows 8 apps

Note: This tip applies only to Windows 8 installations on remote computers.

Because many Windows 8 apps have dynamic user interfaces, avoid using these apps for image-based synchronization.

Working with Clipboard Data (RDP Protocol)

Note: This topic applies to RDP Vuser scripts only.

VuGen allows you to copy and paste the text of a clipboard during an RDP session. You can copy the contents locally and paste them remotely, or vice versa—copy the contents from the remote machine and paste them locally. The copying of text is supported in TEXT, LOCALE, and UNICODE formats.

VuGen generates separate functions when copying or saving the clipboard data.

Code sample #1

The following example illustrates a copy operation on a local machine and a paste operation on a remote machine:

```
//Notifies the Remote Desktop that new data is available in the Local machine's
//clipboard.The data can be provided in three formats: TEXT, UNICODE and LOCALE
rdp_notify_new_clipboard_data(
"StepDescription=Send local clipboard formats 1",
"Snapshot=snapshot1.inf",
"FormatsList=RDP_CF_TEXT|RDP_CF_UNICODE|RDP_CF_LOCALE",
RDP_LAST );
rdp_key(
"StepDescription=Key Press 2",
"Snapshot=snapshot_9.inf",
"KeyValue=V",
"KeyModifier=CONTROL_KEY",
RDP_LAST );
//Provides clipboard data to the Remote Desktop when it requests the data.
rdp_send_clipboard_data(
"StepDescription=Set Remote Desktop clipboard 1",
"Snapshot=snapshot1.inf",
"Timeout=20",
REQUEST_RESPONSE, "Format=RDP_CF_UNICODE", "Text=text for clipboard",
RDP_LAST);
```

Code sample #2

This example illustrates a copy operation on a remote machine and a paste operation on a local machine:

```
rdp_key(
"StepDescription=Key Press 2",
"Snapshot=snapshot_9.inf",
"KeyValue=C",
"KeyModifier=CONTROL_KEY",
RDP_LAST);
// The function requests the Remote Desktop UNICODE text and saves it to a
//parameter
rdp_receive_clipboard_data(
"StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot1.inf",
"ClipboardDataFormat=RDP_CF_UNICODE",
"ParamToSaveData=MyParam",
RDP_LAST);
```

Normally, the Remote Desktop clipboard data is saved in UNICODE format. If the Remote Desktop requests data in the TEXT or LOCALE formats, the **rdp_send_clipboard_data** function automatically converts the content of MyParam from UNICODE into the requested format and sends it to the Remote

Desktop. The Replay log indicates this conversion with an informational message. If the conversion is not possible, the step fails.

For more information about the rdp functions, see the [Function Reference \(Help > Function Reference\)](#).

Correlating Clipboard Parameters

During a recording session, if the client sends the server the same data that it received, VuGen replaces the sent data with a parameter during code generation. VuGen performs this correlation only when the received and sent data formats are the same.

Code sample #3

The following example shows how the same parameter, **MyParam**, is used for both receiving and sending the data.

```
// Receive the data from the server
rdp_receive_clipboard_data("StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot_9.inf",
"Timeout=0",
"ClipboardDataFormat=RDP_CF_UNICODETEXT",
"ParamToSaveData=MyParam",
RDP_LAST);
...
// Send the data to the server
rdp_send_clipboard_data("StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot_9.inf",
"Timeout=10",
REQUEST_RESPONSE, "Format=RDP_CF_UNICODETEXT", "Text={MyParam}",
RDP_LAST);
```

RDP Snapshots - Overview

Note: This topic applies to RDP Vuser scripts only.

Vuser scripts based on the RDP protocol utilize VuGen's Snapshot pane.

- For details on how to work with the Snapshot pane, see ["Work with Snapshots" on page 279](#).
- For details on the Snapshot pane UI, see ["Snapshot Pane" on page 62](#).

When you open an RDP Vuser script, VuGen's standard Snapshot pane functionality is available. The Snapshot pane displays snapshots of the remote display, saved during recording and playback of the Vuser script. Typically, these snapshots are used to synchronize playback of the Vuser script.

In addition to the basic Snapshot pane functionality, the Snapshot pane for RDP Vuser scripts lets you display snapshots in one of the following views:

- **Image.** Displays only the image of the snapshot and is ideal for visually comparing two images. This view displays the snapshot faster and requires less memory than the Full view. You can synchronize two snapshots in the Snapshot pane if both snapshots are displayed in the Image view. The Image view does not automatically scroll to show the area of a snapshot that is used for synchronization.
- **Full.** Scrolls to display the area that is used for synchronization. This view displays the snapshot slower and requires more memory than the Image view. You cannot synchronize two snapshots displayed in the Snapshot pane if either of the snapshots is displayed in the Full view. By default, snapshots are displayed in the Full view.

To display a specific synchronization snapshot in the Snapshot pane, do one of the following:

- In the Editor, select the step that contains a reference to the snapshot.
- In the Step Navigator, double-click the step that contains a reference to the snapshot.

When working with RDP Vuser scripts, the Snapshot pane lets you copy a snapshot to the clipboard, and display a snapshot of the most recent replay error. For more information on how to use the Snapshot pane, see ["Work with Snapshots" on page 279](#).

Image Synchronization Overview (RDP)

An RDP session executes remotely on a computer that is referred to as the server. All keyboard and mouse operations are done on the server, and it is the server that reacts to input from the keyboard and mouse. For example, when you double-click an application icon on the desktop, it is the server that realizes that a double-click took place, and that the application must be loaded.

When an RDP client connects to a server, the client does two things:

- It sends the server coordinates of actions. For example, 'clicked the left mouse button at coordinates (100, 100) on the screen'.
- It receives images from the server showing the current status of the screen after the action took place.

The RDP client (and therefore, the Vuser) does not know that the remote screen contains windows, buttons, icons, and other objects. It knows only that the screen contains an image and at what coordinates the user performed an action. To allow the server to correctly interpret an action, you set a synchronization point within the script. When you replay the script, the synchronization point instructs the Vuser to wait until the image on the server screen matches the corresponding image stored as part of the synchronization point.

See also:

- ["Image Synchronization Tips \(RDP Protocol\)" below](#)

Image Synchronization Tips (RDP Protocol)

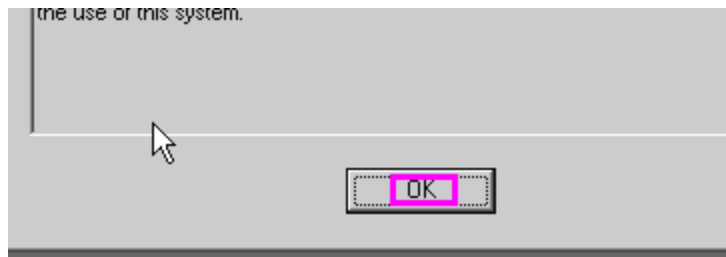
Note: This topic applies to RDP Vuser scripts only.

Use the following guidelines for effective image synchronization:

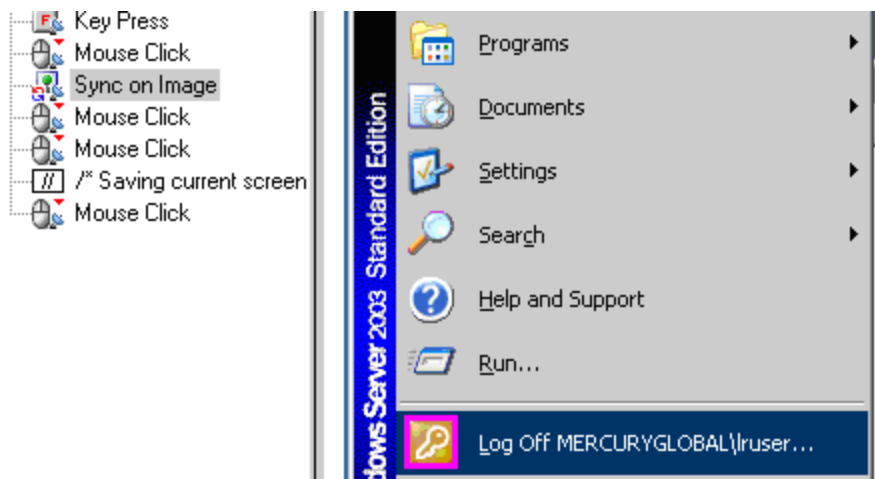
Synchronize on the Smallest Significant Area

When synchronizing on an image, try to synchronize only the part of the image that is necessary. Additional details within the image may not be reproduced during replay and could result in a synchronization failure.

For example, when synchronizing on an image of a button, select only the text itself and not the dotted lines around the text as they may not appear during replay.



When synchronizing a highlighted area, try to capture only the part of the image that is not effected by the highlighting. In the following example, perform a synchronization on the Log Off icon, but not the entire button, since the highlighting may not appear during replay, and the color could vary with different color schemes.



Synchronize Before Every User Action

It is recommended that you synchronize before every mouse operation. You should also synchronize before the first **rdp_key** or **rdp_type** operation that follows a mouse operation.

Image Synchronization - Shifted Coordinates (RDP Protocol)

Note: This topic applies to RDP Vuser scripts only.

When replaying a script, a recorded object may appear at different coordinates on the screen. The object is the same, but its placement has been shifted. For example, during recording a window opened at coordinates (100, 100), but during replay at (200, 250).

In this case, the synchronization point will automatically find the new coordinates without any intervention on your part. It will automatically note the difference of 100 pixels in the horizontal axis and 150 pixels in the vertical axis.

All subsequent mouse operations that are coordinate dependent will use the modified coordinates, so that a mouse click recorded at (130, 130) will be replayed to (230, 280) = (130 + 100, 130 + 150).

You control the shifting of the coordinates through the **AddOffsetToInput** parameter in the **rdp_sync_on_image** step. You can override this parameter to either add or not add the differences in location during replay to the recorded coordinates for any further operations. If you do not override this parameter, VuGen takes its value from the default setting in the runtime settings.

The corresponding parameter in the operations (for example **rdp_mouse_click** or **rdp_mouse_drag**) is **Origin**. This parameter decides whether the operation should take its coordinates only from the 'clean' values that were recorded, or whether it should take into account the differences that were added by the last synchronization point. If not explicitly specified, VuGen takes the value for this parameter from the runtime settings.

Setting Security Levels in RDP Vuser Scripts

Note: This topic applies to RDP Vuser scripts only.

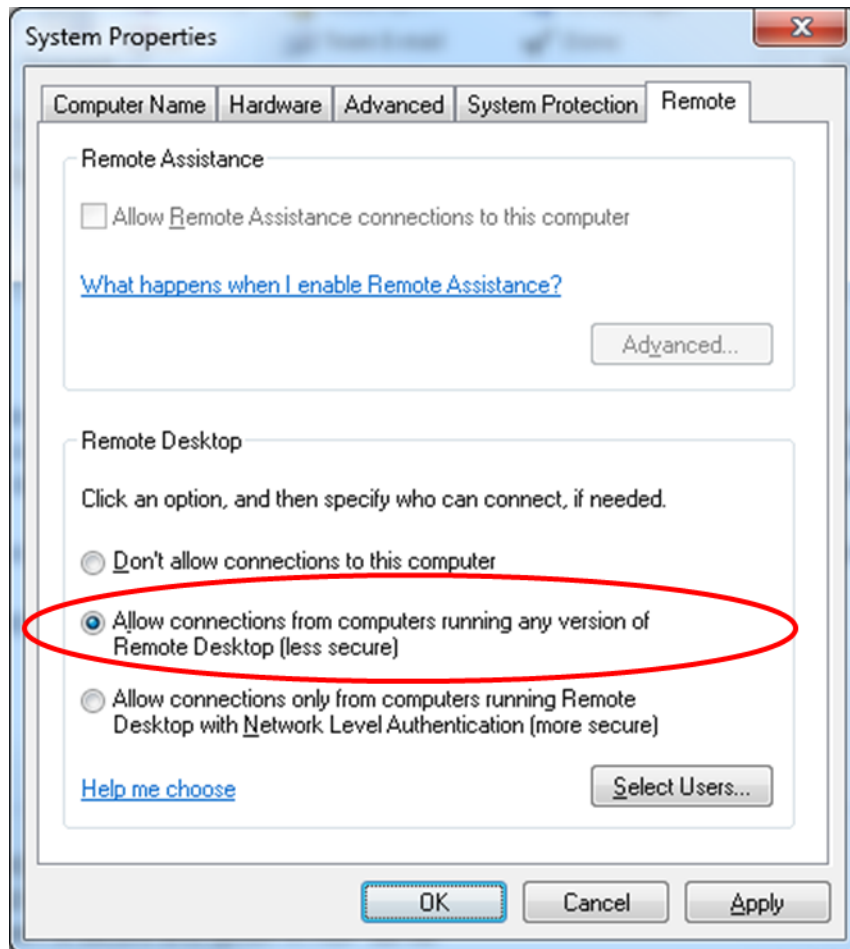
Remote Desktop Protocol (RDP) enables a client computer to connect to a server. Various security options are available for the connection, depending on the particular Windows operating systems that are installed on the client and server computers. The security options define security-related issues, such as the authentication and encryption, that are used for the connection.

The list of security options that are available for a Vuser script is different when you record a Vuser script and when you replay the script.

Security levels when recording an RDP Vuser Script

Standard RDP security is the only form of security that you can use when you record an RDP Vuser script. Before you record an RDP Vuser script, make sure that the server is configured to allow connections from computers that are running any version of Remote Desktop, and not only from computers that are running Remote Desktop with Network Level Authentication. You use the **Remote** tab in the System Properties dialog box on the server to set the security level that is required to establish the connection.

Note: If your RDP session is launched through an RDP configuration file, you must disable *credssp authentication* in the configuration file, using the following string:
`enablecredsspsupport:i:0`



Security levels when replaying an RDP Vuser Script

You can use the Vuser script's runtime settings to specify the security that is used for the connection when the Vuser runs. The available security levels are:

- **RDP:** Connects using standard RDP security. RDP provides the least secure connection.
- **SSL:** Connects using SSL as an external security protocol to enhance the standard RDP security. SSL provides a moderate level of security.
- **CredSSP:** Connects using the Credential Security Support Provider (CredSSP) protocol. CredSSP provides the most secure connection.

Note: If you specify CredSSP authentication, you must make certain changes to the Vuser script each time the script is regenerated. For details, see [Modifying a script to support CredSSP authentication](#) below.

The security level that you specify in the runtime settings is an indication to the server of the maximum level of security that is supported by the client. However, the security that is actually used for the connection is defined by the server settings. For example, if you specify CredSSP as the encryption level in the runtime settings, when you run the Vuser, the Vuser will inform the server that the Vuser

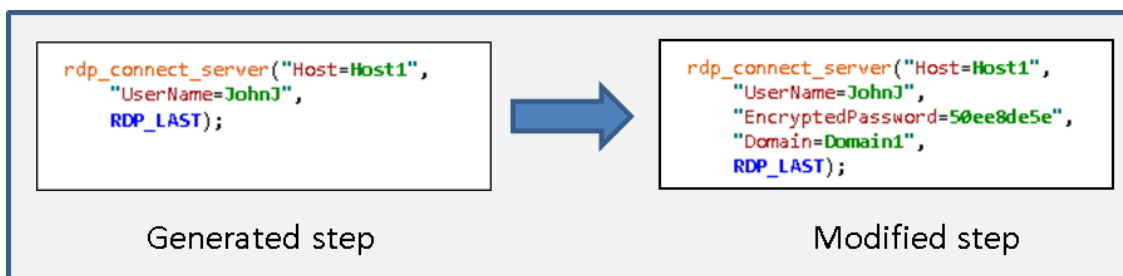
supports CredSSP, SSL, and RDP security. If the server supports only RDP security (for example, its operating system is Windows 2003), then the connection will be made using RDP.

To set the RDP security level for the Vuser script, click **Replay > Runtime Settings > RDP > Configuration** and then select the required level from the **Supported Encryption Level** list.

Modifying a script to support CredSSP authentication

If you specify CredSSP authentication in the Vuser script's runtime settings, you must perform the following tasks each time the script is regenerated:

1. In the **rdp_connect_server** step in the script, modify the step to provide the user name, password, and domain that are required to access the server. For details on the **rdp_connect_server** step, see the Function Reference (**Help > Function Reference**).



2. Remove the block of code that contains the login-related mouse, keyboard, and image synchronization steps from the generated script, as described below.
 - a. Locate the **rdp_connect_server** step in the Vuser script.

The step after the **rdp_connect_server** step is the first step in the block of code to delete.
 - b. Locate the **rdp mouse_click** step or the **rdp_key step** that submits the password to the server.

This is the last step in the block of code to delete.

Note: If an **rdp_set_lock** step exists immediately after the **rdp_connect_server** step, do not delete the **rdp_set_lock** step.

- c. Delete all the steps in the block of code that is defined above.

Modify this step.	<code>rdp_connect_server("Host=Host1", "UserName=John1", "EncryptedPassword=50ee8de5e", "Domain=Domain1", RDP_LAST);</code>
Do not delete this step.	<code>rdp_set_lock("StepDescription=Lock Key Set 1", "LockKeyValue=VK_LIRLOCK", RDP_LAST);</code>
Delete the code that is shaded in yellow.	<code>lr_think_time(12); rdp_sync_on_image("StepDescription=Image Synchronization 1", "WaitFor=Appear", "AddOffsetToInput=Default", IMAGEDATA, "ImageLeft=644", "ImageTop=562", "ImageWidth=40", "ImageHeight=40", "ImageName=snapshot_2.png", ENDIIMAGE, RDP_LAST); rdp_mouse_click("StepDescription=Mouse Click 1", "Snapshot=snapshot_1.inf", "HouseX=664", "HouseY=582", "HouseButton=LEFT_BUTTON", "Origin=Default", RDP_LAST); rdp_type("StepDescription=Typed Text 1", "Snapshot=snapshot_3.inf", "TypedKeys=Password1", RDP_LAST); lr_think_time(11); rdp_sync_on_image("StepDescription=Image Synchronization 2", "WaitFor=Appear", "AddOffsetToInput=Default", IMAGEDATA, "ImageLeft=958", "ImageTop=616", "ImageWidth=40", "ImageHeight=40", "ImageName=snapshot_5.png", ENDIIMAGE, RDP_LAST); rdp_mouse_click("StepDescription=Mouse Click 2", "Snapshot=snapshot_4.inf", "HouseX=978", "HouseY=636", "HouseButton=LEFT_BUTTON", "Origin=Default", RDP_LAST);</code>
The password is entered in this step.	
The password is submitted in this step.	

RDP Agent (for Microsoft Terminal Server) Overview

The Agent for Microsoft Terminal Server is an optional utility that you can install on the RDP server. It provides enhancements to the normal RDP functionality. It is provided in the LoadRunner installation DVD and you can install it on any RDP server. The agent provides you with more intuitive and readable scripts, built-in synchronization, and detailed information about relevant objects. Note that when you run RDP Vusers with the agent installed, each Vuser runs its own process of `lrrdpagent.exe`. This results in a slight reduction in the number of Vusers that can run on the server machine.

Tips for Using the Agent for Microsoft Terminal Server

- When opening application menus (e.g. File, Edit...) with the mouse, sync steps will sometimes fail. To avoid this issue, use the keyboard to select menu items when recording.
- When you add a **rdp_sync_object_mouse_click** step manually, the coordinates given are absolute coordinates (relating to the entire screen). To create the synchronization point, you need to calculate the offset in the window (relative coordinates) of the desired click location and modify the absolute coordinates accordingly for the synchronization to successfully replay.
- If a synchronization object exists at the correct location and time during replay, but is covered by another window (such as a pop-up), then the synchronization step will pass and a click will be executed on the window which is covering the synchronization point and therefore harm the script flow.

- During recording, if you want to return the application window to the foreground, either click on the title bar, or use the keyboard (ALT+TAB). Note that if you click inside the application window to return it to the foreground, the RDP session may terminate unexpectedly.

Enhancements to RDP functionality

The Agent for Microsoft Terminal Server provides the following enhancements to the normal RDP functionality:

Resumed Sessions

Beginning with LoadRunner version 12.50, the RDP Agent process is always active on the server machine. This allows you to record a resumed RDP session on a machine with the RDP agent—you do not need to start a new one.

Object Detail Recording

When the Agent for Microsoft Terminal Server is installed, VuGen can record specific information about the object that is being used instead of general information about the action. For example, VuGen generates **rdp_sync_object_mouse_click** and **rdp_sync_object_mouse_double_click** steps instead of **rdp_mouse_click** and **rdp_mouse_double_click** that it generates without the agent.

The following example shows a double-mouse-click action recorded with and without the agent installation. Note that with the agent, VuGen generates sync_object functions for all of the mouse actions.

```
rdp_sync_object_mouse_double_click("StepDescription=Mouse Double Click on  
Synchronized Object 1",  
    "Snapshot=snapshot_12.inf",  
    "WindowTitle=RDP2",  
    "Attribute=TEXT",  
    "Value=button1",  
    "MouseX=100",  
    "MouseY=71",  
    "MouseButton=LEFT_BUTTON",  
    RDP_LAST);  
rdp_mouse_double_click("StepDescription=Mouse Double Click 1",  
    "Snapshot=snapshot_2.inf",  
    "MouseX=268",  
    "MouseY=592",  
    "MouseButton=LEFT_BUTTON",  
    "Origin=Default",  
    RDP_LAST);
```

Expanded Right-Click Menu

When you click within a snapshot, you can insert several functions into the script using the right-click menu. When the agent is not active, you are limited to inserting only **rdp_mouse_click**, **rdp_mouse_**

double_click, and **rdp_sync_on_image** steps. When the agent is installed, you are able to insert all possible steps that involve the RDP agent:

- **rdp_get_object_info** and **rdp_sync_on_object_info**. Provide information about the state of the object, and synchronize on a specific object property such as: ENABLED, FOCUSED, CONTROL_ID, ITEM_TEXT, TEXT, CHECKED, and LINES.
- **rdp_sync_on_text** and **rdp_get_text**. For details, see the Function Reference (**Help > Function Reference**).

In the following example, the **rdp_sync_on_object_info** function provides synchronization by waiting for the Internet Options dialog box to come into focus.

```
rdp_sync_on_object_info("StepDescription=Sync on Object Info 0",  
    "Snapshot=snapshot_30.inf",  
    "WindowTitle=Internet Options",  
    "ObjectX=172",  
    "ObjectY=155",  
    "Attribute=FOCUSED",  
    "Value={valueParam}",  
    "Timeout=10",  
    "FailStepIfNotFound=No",  
    RDP_LAST);
```

Install/Uninstall the RDP Agent

Note: This topic applies to RDP Vuser scripts only.

If you are upgrading the agent, make sure to uninstall the previous version before installing the new one (see uninstallation instructions below).

Install the RDP Agent (Agent for Microsoft Terminal Server)

1. If your server requires administrator permissions to install software, log in as an administrator to the server.
2. Locate the installation file, **Setup.exe**, on the LoadRunner DVD in the **Additional Components\Agent for Microsoft Terminal Server** folder.
3. Install the RDP agent on your RDP server machine (not on Load Generator machines), following the installation wizard to completion.

Use the RDP Agent (Agent for Microsoft Terminal Server)

Set the recording options before recording a Vuser script.

1. In the Start Recording dialog box, click **Options**.
2. In the Advanced Code Generation node, select the **Use RDP Agent** check box.

Uninstall the RDP Agent (Agent for Microsoft Terminal Server)

1. If your server requires administrator privileges to remove software, log in to the server as an administrator.
2. Select **Control Panel > Add/Remove Programs > HPE Software Agent for Microsoft Terminal Server** and click **Change/Remove**.

Add Image Synchronization Points to a Script

Note: This topic applies to RDP Vuser scripts only.

1. Select an operation to which you would like to add a synchronization point in your script.
2. Right-click on the image snapshot and select **Insert Synch On Image** from the menu. The cursor will change to a cross-hair.
3. Mark the area on the screen that you would like to synchronize upon by clicking on the left button and dragging the box to enclose the area. When you release the mouse button, the **Sync on Image** dialog box opens.
4. Click **OK**. VuGen adds a new **Sync on Image** step before the selected step. When you select this step, VuGen displays a snapshot that contains a pink box around the area you selected for synchronization.

The next time you replay the script, it will wait until the image returned by the server matches the image you selected.

Failed Image Synchronization Dialog Box (RDP Protocol)



This dialog box opens when an image synchronization fails during the replay of a script. You can stop the script or continue the replay despite the error.

The content of this dialog box varies depending on the reason for the failed synchronization:

- **Append Snapshot.** The Failed Image Synchronization - Append Snapshot dialog box opens when the replay image is so different from the record image that changing the tolerance level will not help.
- **Raise Tolerance.** The Failed Image Synchronization - Raise Tolerance dialog box opens when the script replay failed to find the exact image requested, but if the tolerance level for performing synchronization on images was relaxed, then it would have succeeded in finding the image.
- **Lower Tolerance.** The Failed Image Synchronization - Lower Tolerance dialog box opens when the script replay fails to meet the **NotAppear** or **Change** conditions. VuGen detected an image match where you expected it not to detect one. If the tolerance level was reduced, the recorded and replay images would not match, and the **NotAppear** or **Change** conditions would be met resulting in a successful replay.
- **Non Specified.** The Failed Image Synchronization dialog box opens when the script replay fails to meet any of the synchronization conditions such as **NotAppear** or **Change**. VuGen did not find another image at the original coordinates that could be appended to the script.

To access	Opens automatically when an image synchronization fails.
See also	"Image Synchronization Overview (RDP)" on page 607

User interface elements are described below:

UI Element	Description
Stop	Consider the mismatch between the snapshots to be an error. This error will be handled like all other errors and halt the execution of the script.
Continue	<p>This button performs different actions depending on the type of dialog box:</p> <ul style="list-style-type: none"> • Append Snapshot. Accept the mismatch. VuGen appends the replay snapshot as a "record" snapshot for the step. In future replays of the step, VuGen uses all existing record snapshots and the appended snapshot as the basis for comparison between screens. If the replay returns any of the record snapshots, the Vuser will not fail. You can view the original and appended snapshots for a step by clicking the navigation arrows   in the Snapshot pane toolbar. • Lower Tolerance. Accept the mismatch and lower the tolerance level so that VuGen permits a smaller mismatch between the record images and those displayed during replay. • Raise Tolerance. Accept the mismatch and raise the tolerance level so that VuGen permits a greater mismatch between the record images and those displayed during replay. • Non-specified. Accept the mismatch, and do not make any changes in the script. Continue script execution despite the mismatch. <div> <p>Note: Raising or lowering the tolerance level from the dialog box changes the level for the current step only. To change the tolerance level for the whole script, change the Default tolerance for image synchronization setting in the Runtime Settings > RDP > Synchronization view.</p> </div>

Troubleshooting and Limitations for RDP

This section describes troubleshooting information for RDP scripts using the Agent for Microsoft Terminal Server.



Tip: For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

- Clipboard sharing supports only short simple textual content.
- When recording with RDP Agent, applications which were developed using CBuilder may not record

properly.

- RDP does not support 32-bit color depth. If recording uses this color depth, VuGen automatically switches to a lower color depth and a "[RDP Analyzer Warning (790: 418)] 32-bit color depth is not supported, switch to lower one". warning log item appears in the Recording Window.
- For Windows 8, we recommend using the Windows key to switch between the Desktop and Startup screen. This reduces the number of generated mouse calls and simplifies debugging.
- When working on a 64-bit Windows 8.1 machine, you must manually install VcDist_x32 for Visual Studio 2012 before installing the RDP agent.
- Replay fails on **rdp_sync_object_mouse_click/double_click** steps.

Workaround 1: Modify RDPAgentCodeGen.cfg file

The **RDPAgentCodeGen.cfg** file can configure VuGen to automatically create an **rdp_sync_on_image** and **rdp_mouse_click** step the next time the script is generated for each **rdp_sync_object_mouse_click/double_click** steps which occur within a given window. To do this, you specify the name of the window, update a variable which counts the total number of windows for which this process occurs, and regenerate the script.

- a. Open the **RDPAgentCodeGen.cfg** file in the **Script Directory > data** folder.
- b. Open the **Step Navigator** and double-click the step that failed.
- c. Copy the name of the window.
- d. In the **RDPAgentCodeGen.cfg** file, increase the value of **NumberOfTitles** by 1.
- e. Add a line as follows:

`WindowTitleX=<name of window>`

where **X** is the new value of **NumberOfTitles**.
- f. Regenerate the script.

Note: The **RDPAgentCodeGen.cfg** file can be used to automatically produce **rdp_sync_on_image** and **rdp_mouse_click** steps in a similar way for **rdp_sync_object_mouse_click/double_click** steps which are specified in different ways as well. Steps can be targeted based on the class attribute of the control. For more information, contact HPE software support.

Workaround 2: Manually Insert a New Step

Manually insert an **rdp_sync_on_image** and **rdp_mouse_click** step for each step that fails. This method is less desirable, since steps added in this way will be lost if the script is regenerated.

- Connecting to a Windows 10 or Windows Server 2016 server

When using the RDP protocol to record a connection to a Windows 10 or Windows Server 2016 server, the following error may appear on the RDP client: "An authentication error occurred."

Resolution

Perform the following procedure on the server:

- a. Open the Local Group Policy Editor by typing "gpedit.msc" into either a Run prompt or the Start menu.
 - b. In the Local Group Policy Editor, select **Local Computer Policy > Computer Configuration > Administrative Templates > Windows Components > Remote Desktop Services > Remote Desktop Session Host > Security**.
 - c. Enable "Require use of specific security layer for remote (RDP) connections" and set the security layer to **"SSL (TLS 1.0)"**. If you are testing a non-SSL connection, set the security layer to **Negotiate**.
- Failed to connect to RDP server on Windows 10 server

When using the RDP protocol to record a connection to a Windows 10 server, the following error may appear on the RDP client: "An authentication error occurred."

Resolution

Perform the following procedure on the server:

- a. Open the Local Group Policy Editor by typing "gpedit.msc" into either a Run prompt or the Start menu.
- b. In the Local Group Policy Editor, select **Local Computer Policy > Computer Configuration > Administrative Templates > Windows Components > Remote Desktop Services > Remote Desktop Session Host > Security**.
- c. Enable "Require use of specific security layer for remote (RDP) connections" and set the security layer to **RDP**.

Note: Alternatively, open **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp** in the Registry Editor, and set the Value data for **SecurityLayer** to **1**.

RTE Protocol

RTE Protocol Overview

An RTE Vuser types character input into a terminal emulator, submits the data to a server, and then waits for the server to respond. For instance, suppose that you have a server that maintains customer information for a maintenance company. Each time a field service representative makes a repair, he accesses the server database by modem using a terminal emulator. The service representative accesses information about the customer and then records the details of the repair that he performs.

You could use RTE Vusers to emulate this case. An RTE Vuser would:

1. Type **60** at the command line to open an application program.
2. Type **F296**, the field service representative's number.
3. Type **NY270**, the customer number.
4. Wait for the word "Details" to appear on the screen. The appearance of "Details" indicates that all

the customer details are displayed on the screen.

5. Type **Changed gasket P249, and performed Major Service** the details of the current repair.
6. Type **Q** to close the application program.

You use VuGen to create RTE Vuser scripts. The script generator records the actions of a human user on a terminal emulator. The script generator records the keyboard input from the terminal window, generates the appropriate statements, and inserts them into the Vuser script. While you record, the script generator automatically inserts synchronization functions into the script. For details, see ["RTE Synchronization Overview" on page 625](#).

The functions developed to emulate a terminal communicating with a server are called TE Vuser functions. Each TE Vuser function has a **TE** prefix. VuGen automatically records most of the TE functions listed in this section during an RTE recording session. You can also manually program any of the functions into your script.

For syntax and examples of the TE functions, see the Function Reference (**Help > Function Reference**).

An RTE Vuser emulates the actions of a real user. Human users use terminals or terminal emulators to operate application programs.



In the RTE Vuser environment, a Vuser replaces the human. The Vuser operates PowerTerm, a terminal emulator.



PowerTerm works like a standard terminal emulator, supporting common protocols such as IBM 3270 =; 5250, VT100, VT220, and VT420-7.

Working with Ericom Terminal Emulation

VuGen supports record and replay with Ericom Terminal Emulators.

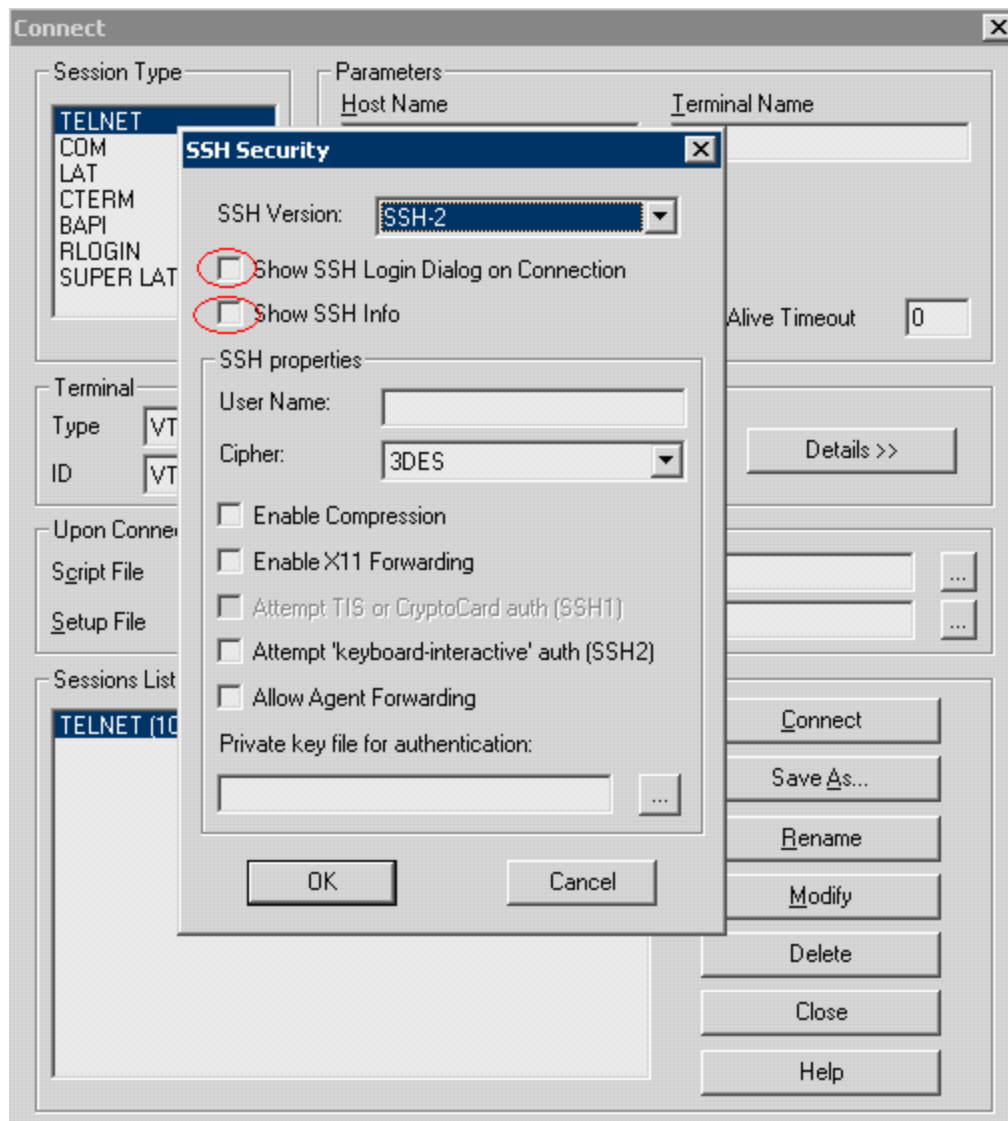
The Ericom support handles escape sequences during record and replay. Ericom's PowerTerm lets you map PC keys to custom escape sequences. For information about mapping, see the PowerTerm help.

When a user presses mapped keys while recording an Ericom VT session, VuGen generates **TE_send_text** functions instead of the standard **TE_type**. This allows the script to handle custom escape sequences in a single step. For more information, see the Function Reference (**Help > Function Reference**) for the **TE_send_text** function.

SSL and SSH Support for Ericom

VuGen also supports SSL/SSH record and replay for the RTE Ericom library. To work with SSL or SSH, you select the type in the **Security** section of the Connect dialog box.

When working with SSH Security, by default VuGen opens a popup dialog box prompting you for more information. We recommend that you disable the **Show** options to prevent the pop-ups from being issued. If you enable these pop-ups, it may affect the replay. You can access the advanced security options by clicking the **Details** button.



Typing Input into a Terminal Emulator

Two TE Vuser functions enable Vusers to "type" character input into the PowerTerm terminal emulator:

- **TE_type** sends characters to the terminal emulator. When recording, the VuGen automatically

generates **TE_type** functions for keyboard input to the terminal window. For details, see below.

- **TE_typing_style** determines the speed at which the Vuser types. You can manually define the typing style by inserting a **TE_typing_style** function into the Vuser script. Alternatively, you can set the typing style by using the runtime settings. For more information, see ["Runtime Settings Overview" on page 283](#).

Note: While recording an RTE Vuser script, do not use the mouse to relocate the cursor within the terminal emulator window. VuGen does not record these cursor movements.

Using the TE_type Function

When you record a script, the VuGen records all keyboard input and generates appropriate **TE_type** functions. During execution, **TE_type** functions send formatted strings to the terminal emulator.

Keyboard input is defined as a regular text string (including blank spaces). For example:

```
TE_type ("hello, world");
```

Input key names longer than one character are represented by identifiers beginning with the letter k, and are bracketed within greater-than/less-than signs (< >).

For example, the following function depicts the input of the Return key followed by the Control and y keys:

```
TE_type("<kReturn><kControl-y>");
```

Some other examples include: <kF1>, <kUp>, <kF10>, <kHelp>, <kTab>.

To determine a key name, record an operation on the key, and then check the recorded statement for its name.

Note: When you program a **TE_type** statement (rather than recording it), use the key definitions provided in the Function Reference (**Help > Function Reference**).

Setting the Timeout Value for TE_type

If a Vuser attempts to submit a **TE_type** statement while the system is in X SYSTEM (or input inhibited) mode, the Vuser will wait until the X SYSTEM mode ends before typing. If the system stays in X SYSTEM mode for more than TE_XSYSTEM_TIMEOUT milliseconds, then the **TE_type** function returns a TE_TIMEOUT error.

You can set the value of TE_XSYSTEM_TIMEOUT by using **TE_setvar**. The default value for TE_XSYSTEM_TIMEOUT is 30 seconds.

Allowing a Vuser to Type Ahead

Under certain circumstances you may want a Vuser to submit a keystroke even though the system is in X SYSTEM (or input inhibited) mode. For example, you may want the Vuser to press the Break key. You use the `TE_ALLOW_TYPEAHEAD` variable to enable the Vuser to submit a keystroke even though the system is in X SYSTEM mode.

Set `TE_ALLOW_TYPEAHEAD` to zero to disable typing ahead, and to any non-zero number to permit typing ahead. You use **TE_setvar** to set the value of `TE_ALLOW_TYPEAHEAD`. By default, `TE_ALLOW_TYPEAHEAD` is set to zero, preventing keystrokes from being sent during X SYSTEM mode.

For more information about the **TE_type** function and its conventions, see the [Function Reference \(Help > Function Reference\)](#).

Setting the Typing Style

You can set two typing styles for RTE Vuser: FAST and HUMAN. In the FAST style, the Vuser types input into the terminal emulator as quickly as possible. In the HUMAN style, the Vuser pauses after typing each character. In this way, the Vuser more closely emulates a human user typing at the keyboard.

You set the typing style using the **TE_typing_style** function. The syntax of the **TE_typing_style** function is:

```
int TE_typing_style (char * style);
```

where style can be FAST or HUMAN. The default typing style is HUMAN. If you select the HUMAN typing style, the format is:

```
HUMAN, delay [, first_delay]
```

The delay indicates the interval (in milliseconds) between keystrokes. The optional parameter first_delay indicates the wait (in milliseconds) before typing the first character in the string. For example,

```
TE_typing_style ("HUMAN, 100, 500");  
TE_type ("ABC");
```

means that the Vuser will wait 0.5 seconds before typing the letter A; it will then wait 0.1 seconds before typing "B" and then a further 0.1 seconds before typing "C".

For more information about the **TE_typing_style** function and its conventions, see the [Function Reference \(Help > Function Reference\)](#).

In addition to setting the typing style by using the **TE_typing_style** function, you can also use the runtime settings. For details, see ["Runtime Settings Overview" on page 283](#).

Generating Unique Device Names

Some protocols, such as APPC, require a unique device name for each terminal that logs on to the system. Using the runtime settings, you can specify that the `TE_connect` function generate a unique 8-character device name for each Vuser, and connect using this name. Although this solves the

requirement for uniqueness, some systems have an additional requirement: The device names must conform to a specific format. See ["Runtime Settings Overview" on page 283](#) for more information.

To define the format of the device names that the `TE_connect` function uses to connect a Vuser to the system, add an **RteGenerateDeviceName** function to the Vuser script. The function has the following prototype:

```
void RteGenerateDeviceName(char buf[32])
```

The device name should be written into **buf**.

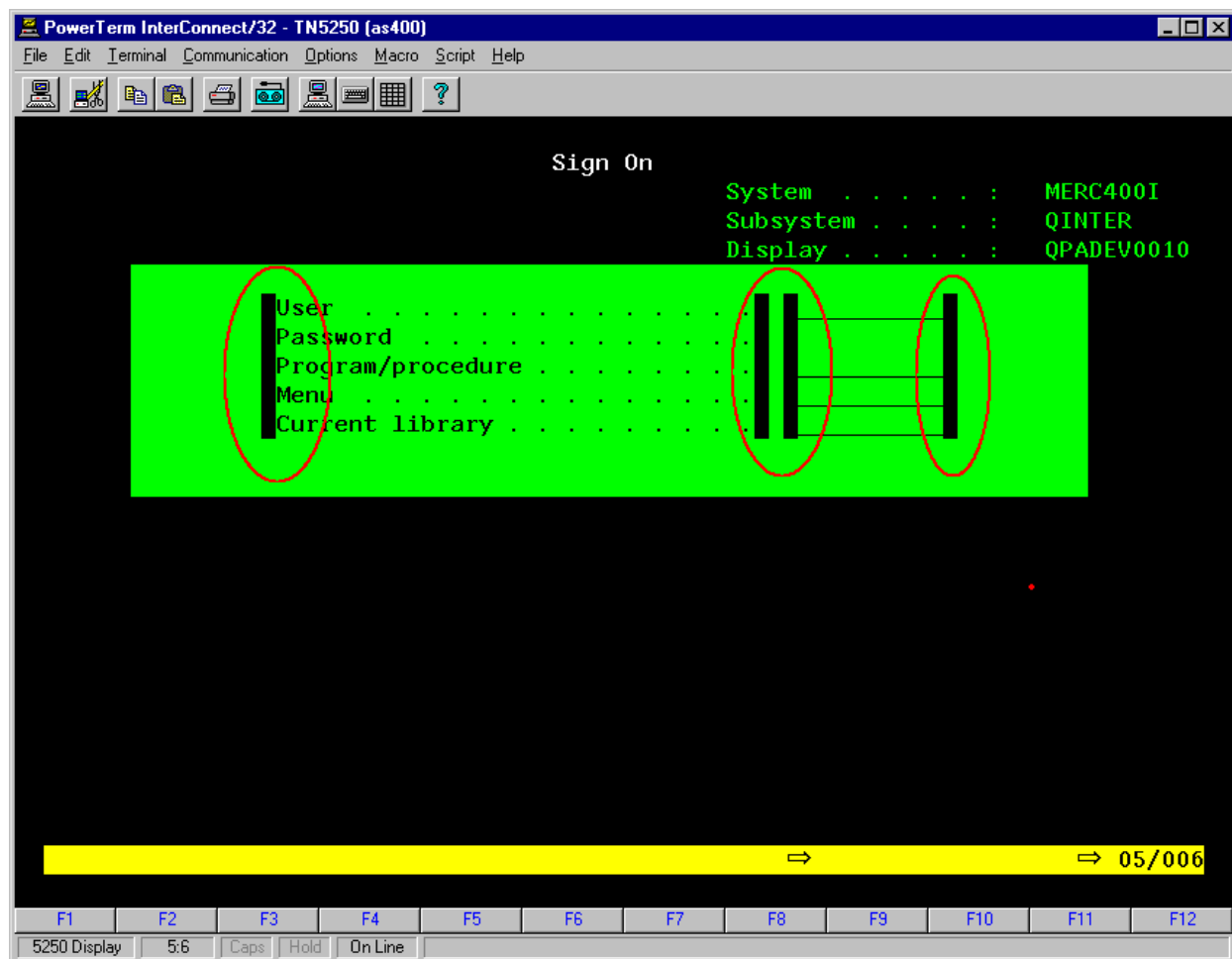
If an **RteGenerateDeviceName** function exists in a Vuser script, the Vuser calls the function each time a new device name is needed. If no **RteGenerateDeviceName** function is defined in the script—and unique device names are required—the **TE_connect** function generates the required names.

In the following example, the `RteGenerateDeviceName` function generates unique device names with the format "TERMx". The first name is TERM0, followed by TERM1, TERM2, and so forth.

```
RteGenerateDeviceName(char buf[32])
{
    static int n=0;
    sprintf(buf, "TERM%d", n);
    n=n+1;
}
```

Setting the Field Demarcation Characters

Some terminal emulators use demarcation characters to mark the beginning and the end of each field. These demarcation characters are not visible—appearing on the screen as spaces. In the terminal emulator shown below, the colors in the middle section of the screen have been inverted to display the field demarcation characters. These characters are surrounded by ellipses.



The **TE_wait_text**, **TE_get_text**, and **TE_find_text** functions operate by identifying the characters in a specified portion of the screen. If a field demarcation character is located within the specified section, you can identify the character as a space or an ASCII character. You use the **TE_FIELD_CHARS** system variable to specify the method of identification. You can set **TE_FIELD_CHARS** to 0 or 1:

- 0 specifies that the character in the position of the field demarcation characters is returned as a space.
- 1 specifies that the character in the position of the field demarcation characters is returned as an ascii code (ascii 0 or ascii 1).

By default, **TE_FIELD_CHARS** is set to 0.

You retrieve and set the value of **TE_FIELD_CHARS** by using the **TE_getvar** and **TE_setvar** functions.

Reading Text from the Terminal Screen

There are several Vuser functions that RTE Vusers can use to read text from the terminal screen. You can use these functions, **TE_find_text** and **TE_get_text_line**, to check that the terminal emulator is responding correctly, or to enhance the logic in your scripts.

After recording, you can manually insert **TE_find_text** and **TE_get_text_line** statements directly into your RTE Vuser scripts.

Searching for Text on the Screen

The **TE_find_text** function searches for a line of text on the screen. The syntax of the function is:

```
int TE_find_text (char *pattern, int col1, int row1, int col2, int row2,  
                 int *retcol, int *retrow, char *match );
```

This function searches for text matching pattern within the rectangle defined by col1, row1, col2, row2. Text matching the pattern is returned to match, and the actual row and column position is returned to retcol and retrow. The search begins in the top-left corner. If more than one string matches pattern, the one closest to the top-left corner is returned.

The **pattern** can include a regular expression. See the Function Reference (**Help > Function Reference**) for details on using regular expressions.

You must manually type **TE_find_text** statements into your Vuser scripts. For details on the syntax of the **TE_find_text** function, see the Function Reference (**Help > Function Reference**).

Reading Text from the Screen

The **TE_get_text_line** function reads a line of text from the area of the screen that you designate. The syntax of the function is:

```
char *TE_get_text_line (int col, int row, int width, char * text );
```

This function copies a line of text from the terminal screen to a buffer text. The first character in the line is defined by col, row. The column coordinate of the last character in the line is indicated by width. The text from the screen is returned to the buffer text. If the line contains tabs or spaces, the equivalent number of spaces is returned.

In addition, the **TE_get_cursor_position** function can be used to retrieve the current position of the cursor on the terminal screen. The **TE_get_line_attribute** function returns the character formatting (for instance, bold or underline) of a line of text.

You must manually type **TE_get_text_line** statements into your Vuser scripts. For details on the syntax of the **TE_get_text_line** function, see the Function Reference (**Help > Function Reference**).

RTE Synchronization Overview

Depending on the system you are testing, you may need to synchronize the input that a Vuser sends to a terminal emulator with the subsequent responses from the server. When you synchronize input, you instruct the Vuser to suspend script execution and wait for a cue from the system, before the Vuser performs its next action. For instance, suppose that a human user wants to submit the following sequence of key strokes to a bank application:

1. Type 1 to select "Financial Information" from the menu of a bank application.
2. When the message "What information do you require?" appears, type 3 to select "Dow Jones

Industrial Average" from the menu.

3. When the full report has been written to the screen, type 5 to exit the bank application.

In this example, the input to the bank application is synchronized because at each step, the human user waits for a visual cue before typing.

This cue can be either the appearance of a particular message on the screen, or stability of all the information on the screen.

You can synchronize the input of a Vuser in the same way by using the TE-synchronization functions, **TE_wait_sync**, **TE_wait_text**, **TE_wait_silent**, and **TE_wait_cursor**. These functions effectively emulate a human user who types into a terminal window and then waits for the server to respond, before typing in the next command.

The **TE_wait_sync** function is used to synchronize block-mode (IBM) terminals only. The other TE-synchronization functions are used to synchronize character-mode (VT) terminals.

When you record an RTE Vuser script, VuGen can automatically generate and insert **TE_wait_sync**, **TE_wait_text**, and **TE_wait_cursor** statements into the script. You use VuGen's recording options to specify which synchronization functions VuGen should insert.

Note: Do not include any synchronization statements in the Vuser_end section of a Vuser script. Since a Vuser can be aborted at any time, you cannot predict when the Vuser_end section will be executed.

Synchronizing Block-Mode (IBM) Terminals

The **TE_wait_sync** function is used for synchronization RTE Vusers operating block-mode (IBM) terminals. Block-mode terminals display the "X SYSTEM" message to indicate that the system is in Input Inhibited mode. When a system is in the Input Inhibited mode no typing can take place because the terminal emulator is waiting for a transfer of data from the server.

When you record a script on a block-mode terminal, by default, VuGen generates and inserts a **TE_wait_sync** function into the script each time the "X SYSTEM" message appears. You use VuGen's recording options to specify whether or not VuGen should automatically insert **TE_wait_sync** functions.

When you run a Vuser script, the **TE_wait_sync** function checks if the system is in the X SYSTEM mode. If the system is in the X SYSTEM mode, the **TE_wait_sync** function suspends script execution. When the "X SYSTEM" message is removed from the screen, script execution continues.

Note: You can use the **TE_wait_sync** function only with IBM block-mode terminals emulators (5250 and 3270).

In general, the **TE_wait_sync** function provides adequate synchronization for all block-mode terminal emulators. However, if the **TE_wait_sync** function is ineffective in a particular situation, you can enhance the synchronization by including a **TE_wait_text** function. For more information on the **TE_wait_text**

function, see ["Synchronizing Character-Mode \(VT\) Terminals" on the next page](#), and the [Function Reference \(Help > Function Reference\)](#).

In the following script segment, the Vuser logs on with the user name "QUSER" and the password "HPLAB". The Vuser then presses Enter to submit the login details to the server. The terminal emulator displays the X SYSTEM message while the system waits for the server to respond.

The **TE_wait_sync** statement causes the Vuser to wait until the server has responded to the login request, that is, for the X SYSTEM message to be removed—before executing the next line of the script.

```
TE_type("QUSER");  
lr_think_time(2);  
TE_type("<kTab>HPLAB");  
lr_think_time(3);  
TE_type("<kEnter>");  
TE_wait_sync();  
....
```

When a **TE_wait_sync** function suspends the execution of a script while an X SYSTEM message is displayed, the Vuser continues to monitor the system—waiting for the X SYSTEM message to disappear. If the X SYSTEM message does not disappear before the synchronization timeout expires, the **TE_wait_sync** function returns an error code. The default timeout is 60 seconds.

Set the **TE_wait_sync** synchronization timeout

1. Select **Vuser > Runtime Settings**. The runtime settings dialog box appears.
2. Select the **RTE:RTE** node in the Runtime setting tree.
3. Under **X SYSTEM Synchronization**, enter a value (in seconds) in the **Timeout** box.
4. Click **OK** to close the runtime settings dialog box.

After a Vuser executes a **TE_wait_sync** function, the Vuser waits until the terminal is no longer in the X SYSTEM mode. When the terminal returns from the X SYSTEM mode, the Vuser continues to monitor the system for a short period to verify that the terminal is fully stable, that is, that the system does not return to the X SYSTEM mode. Only then does the **TE_wait_sync** function terminate and allow the Vuser to continue executing its script. The period that the Vuser continues to monitor the system, after the system has returned from the X SYSTEM mode, is known as the stable time. The default stable time is 1000 milliseconds.

You may need to increase the stable time if your system exhibits the following behavior:

When a system returns from the X SYSTEM mode, some systems "flickers" to and from the X SYSTEM for a short period of time until the system stabilizes. If the system remains out of the X SYSTEM mode for more than one second, and then returns to the X SYSTEM mode, the **TE_wait_sync** function will assume that the system is stable. If a Vuser then tries to type information to the system, the system will shift into keyboard-locked mode.

Alternatively, if your system never flickers when it returns from the X SYSTEM mode, you can reduce the stable time to less than the default value of one second.

Change the stable time for TE_wait_sync functions

1. Select **Vuser > Runtime Settings**. The runtime settings dialog box appears.
2. Select the **RTE:RTE** node.
3. Under **X SYSTEM Synchronization**, enter a value (in milliseconds) in the **Stable time** box.
4. Click **OK** to close the runtime settings dialog box.
For more information on the **TE_wait_sync** function, see the Function Reference (**Help > Function Reference**).

You can instruct VuGen to record the time that the system remains in the X SYSTEM mode each time that the X SYSTEM mode is entered. To do so, VuGen inserts a **TE_wait_sync_transaction** function after each **TE_wait_sync** function, as shown in the following script segment:

```
TE_wait_sync();  
TE_wait_sync_transaction("syncTrans1");
```

Each **TE_wait_sync_transaction** function creates a transaction with the name "default." This allows you to analyze how long the terminal emulator waits for responses from the server during a scenario run. You use the recording options to specify whether VuGen should generate and insert **TE_wait_sync_transaction** statements.

Instruct VuGen to insert TE_wait_sync_transaction statements

1. Select **Vuser > Recording Options**. The Recording Settings dialog box appears.
2. Select the **Generate Automatic X SYSTEM transactions** option, and then click **OK**.

Synchronizing Character-Mode (VT) Terminals

You select various types of synchronization for character-mode (VT) terminals according to:

- the design of the application that is running in the terminal emulator
- the specific action to be synchronized

Wait for the cursor to appear at a specific location

The preferred method of synchronization for VT type terminals is cursor synchronization. Cursor synchronization is particularly useful with full-screen or form-type applications, as opposed to scrolling or TTY-type applications.

Cursor synchronization uses the **TE_wait_cursor** function. When you run an RTE Vuser script, the **TE_wait_cursor** function instructs a Vuser to suspend script execution until the cursor appears at a specified location on the screen. The appearance of the cursor at the specified location means that the application is ready to accept the next input from the terminal emulator.

The syntax of the **TE_wait_cursor** function is:

```
int TE_wait_cursor (int col, int row, int stable, int timeout );
```

During script execution, the **TE_wait_cursor** function waits for the cursor to reach the location specified by col, row.

The **stable** parameter specifies the time (in milliseconds) that the cursor must remain at the specified location. If you record a script using VuGen, **stable** is set to 100 milliseconds by default. If the client application does not become stable in the time specified by the **timeout** parameter, the function returns TIMEOUT. If you record a script using VuGen, **timeout** is set by default to the value of TIMEOUT, which is 90 seconds. You can change the value of both the **stable** and **timeout** parameters by directly editing the recorded script.

The following statement waits for the cursor to remain stable for three seconds. If the cursor doesn't stabilize within 10 seconds, the function returns TIMEOUT.

```
TE_wait_cursor (10, 24, 3000, 10);
```

For more information on the **TE_wait_cursor** function, see the [Function Reference \(Help > Function Reference\)](#).

You can instruct VuGen to automatically generate TE_wait_cursor statements, and insert them into a script, while you record the script. The following is an example of a TE_wait_cursor statement that was automatically generated by VuGen:

```
TE_wait_cursor(7, 20, 100, 90);
```

Instruct VuGen to automatically generate and insert TE_wait_cursor statements while recording

1. Select **Vuser > Recording Options**. The Recording Settings dialog box appears.
2. Under **Generate Automatic Synchronization Commands** select the **Cursor** check box, and then click **OK**.

Wait for text to appear on the screen

You can use text synchronization to synchronize an RTE Vuser running on a VT terminal emulator. Text synchronization uses the **TE_wait_text** function. During script execution, the **TE_wait_text** function suspends script execution and waits for a specific string to appear in the terminal window before continuing with script execution. Text synchronization is useful with those applications in which the cursor does not consistently appear in a predefined area on the screen.

Note: Although text synchronization is designed to be used with character mode (VT) terminals, it can also be used with IBM block-mode terminals. Do not use automatic text synchronization with block-mode terminals.

The syntax of the **TE_wait_text** function is:

```
int TE_wait_text (char * pattern, int timeout, int col1, int row1, int
col2, int row2,
                 int * retcol, int * retrow, char * match );
```

This function waits for text matching pattern to appear within the rectangle defined by col1, row1, col2, row2. Text matching the pattern is returned to **match**, and the actual row and column position is returned to **retcol** and **retrow**. If the **pattern** does not appear before the **timeout** expires, the function returns an error code. The **pattern** can include a regular expression. See the Function Reference for details on using regular expressions. Besides the **pattern** and **timeout** parameters, all the other parameters are optional.

If **pattern** is passed as an empty string, the function will wait for timeout if it finds any text at all within the rectangle. If there is no text, it returns immediately.

If the pattern does appear, then the function waits for the emulator to be stable (finish redrawing, and not display any new characters) for the interval defined by the TE_SILENT_SEC and TE_SILENT_MILLI system variables. This, in effect, allows the terminal to become stable and emulates a human user.

If the terminal does not become stable within the interval defined by TE_SILENT_TIMEOUT, script execution continues. The function returns 0 for success, but sets the TE_errno variable to indicate that the terminal was not silent after the text appeared.

To modify or retrieve the value of any of the TE_SILENT system variables, use the TE_getvar and TE_setvar functions. For more information, see the [Function Reference \(Help > Function Reference\)](#).

In the following example, the Vuser types in its name, and then waits for the application to respond.

```
/* Declare variables for TE_wait_text */
int ret_row;
int ret_col;
char ret_text [80];
/* Type in user name. */
TE_type ("John");
/* Wait for teller to respond. */
TE_wait_text ("Enter secret code:", 30, 29, 13, 1, 13, &ret_col, &ret_row,
             ret_text);
```

You can instruct VuGen to automatically generate **TE_wait_text** statements, and insert them into a script, while you record the script.

Instruct VuGen to automatically generate and insert TE_wait_text statements while recording

1. Select **Vuser > Recording Options**. The Recording Settings dialog box appears.
2. Under **Generate Automatic Synchronization Commands**, select the **Prompt** check box, and then click **OK**.

The following is an example of a TE_wait_text statement that was automatically generated by VuGen. The function waits up to 20 seconds for the string "keys" to appear anywhere on the

screen. Note that VuGen omits all the optional parameters when it generates a `TE_wait_text` function.

```
TE_wait_text("keys", 20);
```

Wait for the terminal to be silent

In instances when neither cursor synchronization nor text synchronization are effective, you can use "silent synchronization" to synchronize the script. With "silent synchronization," the Vuser waits for the terminal emulator to be silent for a specified period of time. The emulator is considered to be silent when it does not receive any input from the server for a specified period of time.

Note: Use silent synchronization only when neither cursor synchronization nor text synchronization are effective.

You use the **TE_wait_silent** function to instruct a script to wait for the terminal to be silent. You specify the period for which the terminal must be silent. If the terminal is silent for the specified period, then the **TE_wait_silent** function assumes that the application has stopped printing text to the terminal screen, and that the screen has stabilized.

The syntax of the function is:

```
int TE_wait_silent (int sec, int milli, int timeout );
```

The **TE_wait_silent** function waits for the terminal emulator to be silent for the time specified by `sec` (seconds) and `milli` (milliseconds). The emulator is considered silent when it does not receive any input from the server. If the emulator does not become silent (i.e. stop receiving characters) during the time specified by the time `timeout` variable, then the function returns an error.

For example, the following statement waits for the screen to be stable for three seconds. If after ten seconds, the screen has not become stable, the function returns an error.

```
TE_wait_silent (3, 0, 10);
```

For more information, see the Function Reference (**Help > Function Reference**).

Map Terminal Keys to PC Keyboard Keys

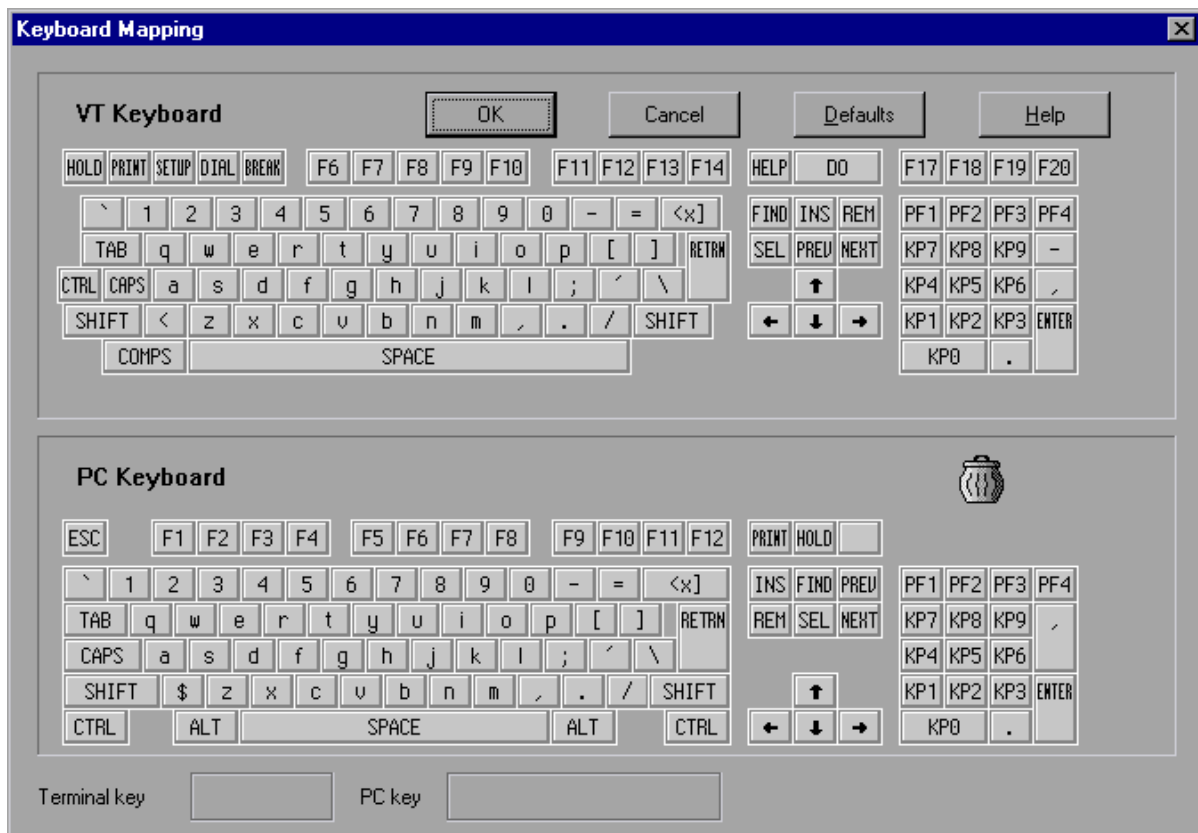
Because you are using a terminal emulator, you will be using a PC keyboard in place of a terminal keyboard. Many keys that are found on the terminal keyboard are not available on a PC keyboard. Examples of such keys are `HELP`, `AUTHOR`, and `PUSH`, which are found on the IBM 5250 keyboard. To successfully operate the terminal emulator and any associated application programs, you may have to map certain terminal keys to keys on the PC keyboard.

Map a Terminal Key to a Key on the PC Keyboard

1. In the terminal emulator, select **Options > Keyboard Map**, or click the **Keyboard Mapping** button



The Keyboard Mapping dialog box opens.



2. Click the **Keyboard Mapping** button on the toolbar. To map a terminal key to a PC key, drag a key from the upper terminal keyboard to a PC key on the lower keyboard.

You can click the Shift and/or Control keys on the upper keyboard to display additional key functions that can be viewed only by first selecting either of these keys. You can then drag the required key from the upper terminal keyboard to a key on the lower PC keyboard.

To cancel a definition, drag the PC key definition to the wastebasket. This restores the default function of the PC key.


To restore the default mappings, click **Defaults**.

Record RTE Vuser Scripts

You use VuGen to record RTE Vuser scripts. VuGen uses the PowerTerm terminal emulator to emulate a wide variety of terminal types.

This task describes how to record RTE Vuser scripts. This procedure differs from the general recording procedure in ["Recording" on page 131](#).

1. Record the terminal setup and connection

- a. Open an existing RTE Vuser script, or create a new one.
- b. In the **Sections** box, select the **vuser_init** section to insert the recorded statements.
- c. In the Vuser script, place the cursor at the location where you want to begin recording.
- d. Click the **Start Record** button  **Start Record**. The PowerTerm main window opens.
- e. From the PowerTerm menu bar, select **Terminal > Setup** to display the Terminal Setup dialog box.
- f. Select the type of emulation from the VT Terminal and IBM Terminal types, and then click **OK**.

Note: Select an IBM terminal type to connect to an AS/400 machine or an IBM mainframe; select a VT terminal type to connect to a Linux workstation.

- g. Select **Communication > Connect** to display the Connect dialog box.
- h. Under **Session Type**, select the type of communication to use.
- i. Under **Parameters**, specify the required options. The available parameters vary depending on the type of session that you select. For details on the parameters, click **Help**.



Tip: Click **Save As** to save the parameter-sets for re-use in the future. The parameter-sets that you save are displayed in the Sessions List box.

- j. Click **Connect**. PowerTerm connects to the specified system, and VuGen inserts a **TE_connect** function into the script, at the insertion point. The **TE_connect** statement has the following form:

```
/* *** The terminal type is VT 100. */
TE_connect(
    "comm-type = telnet;"
    "host-name = alfa;"
    "telnet-port = 992;"
    "terminal-id = ;"
    "set-window-size = true;"
    "security-type = ssl;"
    "ssl-type = tls1;"
    "terminal-type = vt100;"
    "terminal-model = vt100;"
    "login-command-file = ;"
    "terminal-setup-file = ;"
    , 60000);
if (TE_errno != TE_SUCCESS)
    return -1;
```

The inserted **TE_connect** statement is followed by an if statement that checks whether or not the **TE_connect** function succeeds during replay.

Note: Do not record more than one connection to a server (**TE_connect**) in a Vuser script.



2. Record typical user actions

After recording the setup procedure, you perform typical user actions or business processes. You record these processes into the **Actions** section of the Vuser script. Only the **Actions** section of a Vuser script is repeated when you run multiple iterations of the script.

When recording a session, VuGen records the text strokes and not the text. Therefore, it is not recommended that you copy and paste commands into the PowerTerm window—instead, type them in directly.

- a. Select the **Actions** section in the **Section** box.
- b. Proceed to perform typical user actions in the terminal emulator. VuGen generates the appropriate statements, and inserts them into the Vuser script while you type. If necessary, you can edit the recorded statements while you record the script.

3. Record the log-off procedure

- a. Make sure that you have performed and recorded the typical user actions as described in the previous section.
- b. In the VuGen main window, click **vuser_end** in the **Section** box.
- c. Perform the log off procedure. VuGen records the procedure into the **vuser_end** section of the script.
- d. Click **Stop Recording**  on the Recording toolbar. The main VuGen window displays all the recorded statements.
- e. Click  **Save** to save the recorded session. After recording a script, you can manually edit it in VuGen's main window.

Implement Continue on Error

To configure the Continue on Error functionality in RTE Scripts:

- To continue running the script on error, insert the following function:
TE_setvar(TE_IGNORE_ERRORS, 1)
- To restore the default behavior of failing the script on error, insert the following function:
TE_setvar(TE_IGNORE_ERRORS, 0)

Troubleshooting and Limitations - RTE

This section describes troubleshooting and limitations for RTE Vusers.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

IP Spoofing

IP spoofing is not supported for RTE Vusers.

Disconnection Failures

When running an RTE script from the Controller, it may hang in the **Gradual Exiting** stage. A possible reason may be that the terminal session tried to disconnect but did not finish disconnecting before a new connect command arrived. As a result, the scenario cannot exit properly.

Possible workaround: In scripts created with versions of LoadRunner prior to 12.50, **TE_disconnect** was not automatically recorded. Manually add **TE_disconnect** functions to older RTE scripts.

Initialization Failures

When running an RTE script from the Controller, it may fail during initialization with an error message like this:

"Warning: Extension rterun32.dll reports error -1 on call to function ExtPerThreadInitialize. Error: Vuser failed to initialize extension rterun32.dll."

Possible solution: This could be an "Out of Memory" issue. This can be fixed by increasing the Desktop Heap. In this case, change the last value of the **SharedSection** parameter in **[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SubSystems\Windows]**.

For more details on how to fix "Out of memory" issues, see <https://support.microsoft.com/en-us/kb/126962>.

SAP Protocols

Selecting an SAP Protocol Type

- To test the SAP GUI user operating only on the client, use the SAP GUI Vuser type.
- To test a SAP GUI user that also uses a Web browser, use the SAP-Web protocol.

To record a SAP GUI session that uses browser controls, create a multi-protocol Vuser script with the SAP GUI and SAP-Web protocols. This allows VuGen to record Web-specific functions when encountering the browser controls. This will not work if you attempt to combine SAP GUI and Web protocols.

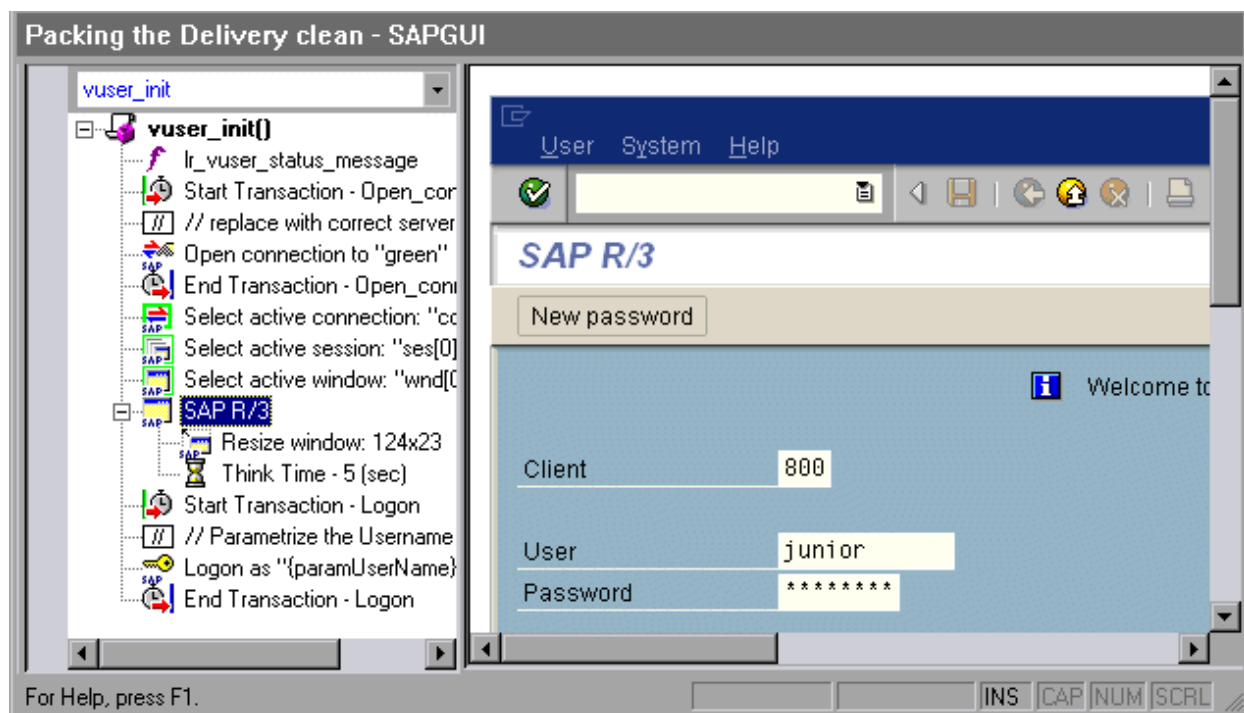
Before recording a session, verify that your modules and client interfaces are supported by VuGen. The following table describes the SAP client modules for SAP Business applications and the relevant tools:

SAP module	VuGen support
SAP Web Client or mySAP.com	Use the SAP-Web protocol.
SAP GUI for Windows	Use the SAP GUI protocol. This also supports APO module recording (requires patch level 24 for APO 3.0 for SAP 6.20).
SAP GUI for Java	This client is not supported.

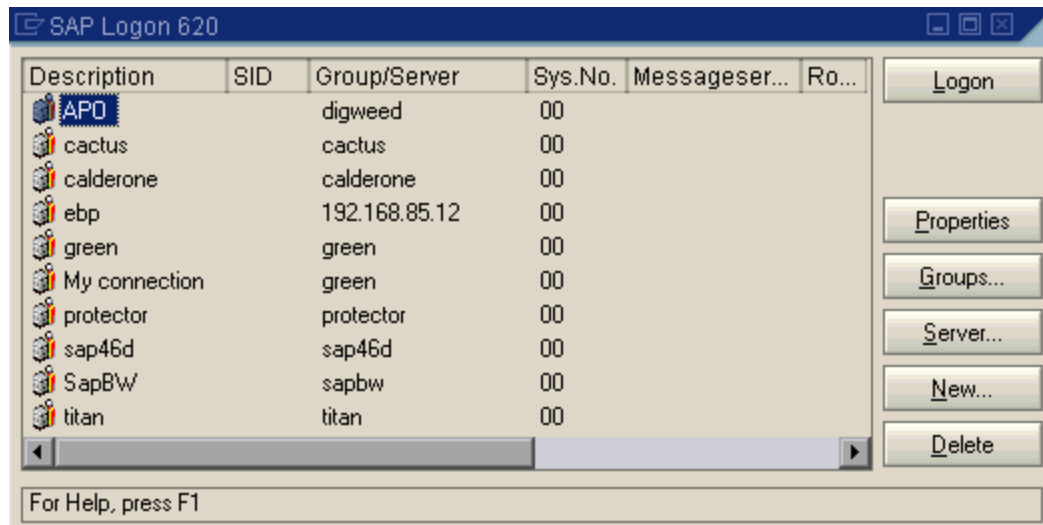
SAP GUI Protocol

The SAP GUI Vuser script typically contains several SAP transactions which make up a business process. A business process consists of functions that emulate user actions. Open the **Step Navigator** to see each user action as a Vuser script step.

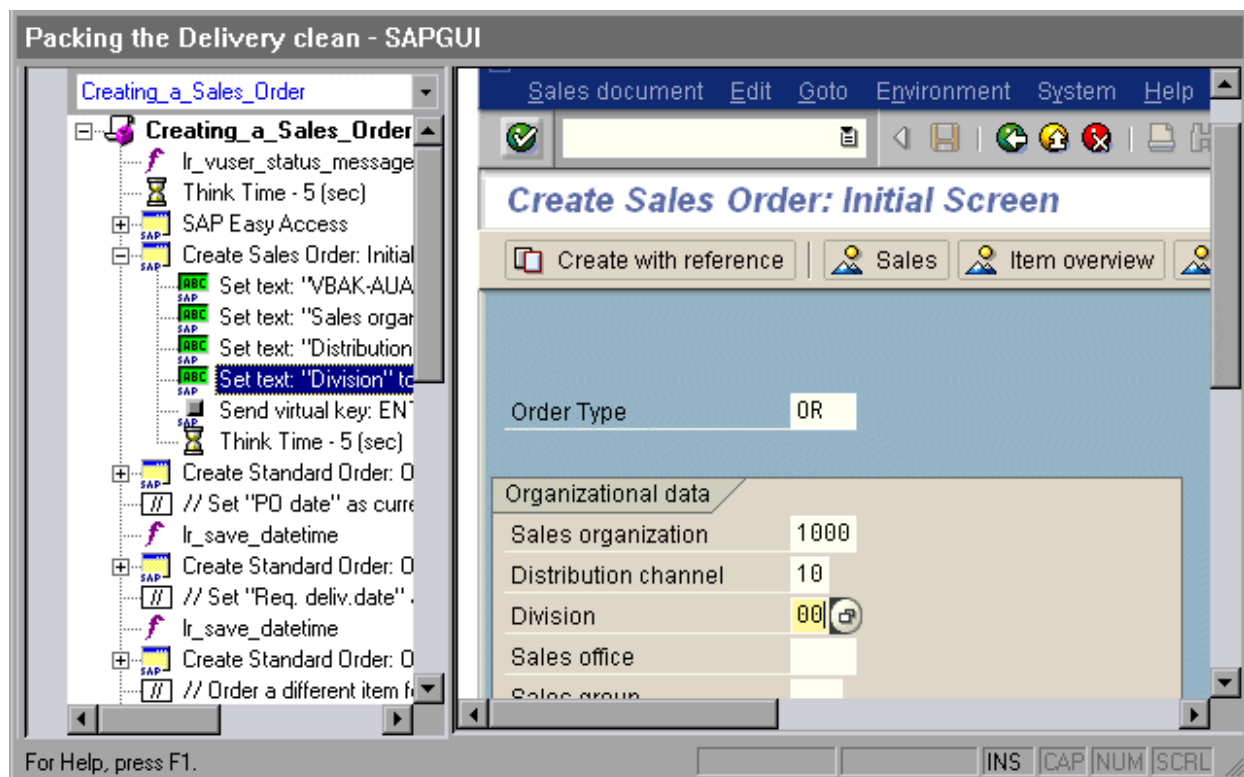
The following example shows a typical recording of a SAP GUI client. The first section, **vuser_init**, contains the opening of a connection and a logon.



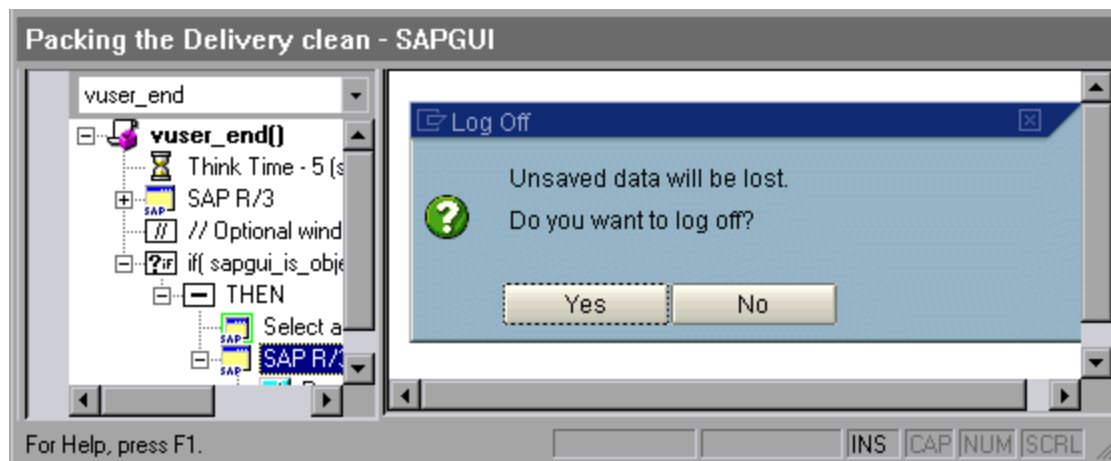
Note: The Open Connection step uses one of the connection names in the SAP Logon **Descriptions** list. If the specified connection name is not in the list, the Vuser looks for a server with that name.



In the following section, the functions emulate typical user operations such as menu selection and the setting of a check box.



The final section, **vuser_end**, illustrates the logoff procedure.



When recording a multi-protocol script for both SAP GUI and Web, VuGen generates steps for both protocols. In the Script view, you can view both **sapgui** and **web** functions.

The following example illustrates a multi-protocol recording in which the SAP GUI client opens a Web control. Note the switch from **sapgui** to **web** functions.



Example:

```
sapgui_tree_double_click_item("Use as general WWW browser, REPTITLE",
    "shellcont/shell",
    "000732",
    "REPTITLE",
    BEGIN_OPTIONAL,
    "AdditionalInfo=sapgui1020",
    END_OPTIONAL);
...
sapgui_set_text("",
    "http:\\\\yahoo.com",
    "usr/txtEDURL",
    BEGIN_OPTIONAL,
    "AdditionalInfo=sapgui1021",
    END_OPTIONAL);
...
web_add_cookie("B=7pt5civ1p3m2=;b=2; DOMAIN=www.yahoo.com");
web_url("yahoo.com",
    "URL=http://yahoo.com/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    EXTRARES,
    "URL=
http://srd.yahoo.com/hpt1/ni=17/ct=lan/sss=1043752588/t1=1043752575385/d1=1251
/d2=1312/d3=1642/d4=4757/0.4097009487287739/*1",
```



```
"Referer=http://www.yahoo.com/", ENDITEM,  
LAST);
```

SAP Web Protocol

The SAP-Web Vuser script typically contains several SAP transactions which make up a business process. The business process consists of functions that emulate user actions. For information about these functions, see the Web functions in the **Function Reference (Help > Function Reference)**.

Note: You can generate a SAP - Web Vuser script by analyzing an existing network traffic file (capture file). This method may be useful for creating Vuser scripts that emulate activity on mobile applications. For details, see ["Create a Vuser Script by Analyzing a Captured Traffic File" on page 685](#).

Example:

The following example shows a typical recording for an SAP Portal client:

```
vuser_init()  
{  
    web_reg_find("Text=SAP Portals Enterprise Portal 5.0",  
        LAST);  
    web_set_user("junior{UserNumber}",  
        lr_unmask("3ed4cfe457afe04e"),  
        "sonata.hplab.com:80");  
    web_url("sapportal",  
        "URL=http://sonata.hplab.com/sapportal",  
        "Resource=0",  
        "RecContentType=text/html",  
        "Snapshot=t1.inf",  
        "Mode=HTML",  
        EXTRARES,  
        "Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/branding_  
image.jpg",  
        "Referer=http://sonata.hplab.com/hrnp$30001/sonata.hplab.com:80/Action/26011[header]  
",  
        , ENDITEM,  
        "Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/logo.gif",  
        "Referer=http://sonata.hplab.com/hrnp$30001/sonata.hplab.com:80/Action/26011[header]  
",  
        ,  
        , ENDITEM,  
        ...  
        LAST);  
}
```

The following section illustrates an SAP Web and SAP GUI multi-protocol recording in which the Portal client opens an SAP control. Note the switch from **web_xxx** to **sapgui_xxx** functions.

```
web_url("dummy",
    "URL=http://sonata.hplab.com:1000/hrnp$30000/sonata.hplab.com:
    1000/Action/dummy?PASS_PARAMS=YES=;dummyComp=dummy=;
Tcode=VA01=;draggable=0=;CompFName=VA01=;Style=sap_mango_polarwind",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=http://sonata.hplab.com/sapportal",
    "Snapshot=t9.inf",
    "Mode=HTML",
    LAST);
sapgui_open_connection_ex(" /H/Protector/S/3200 /WP",
    "",
    "con[0]");
sapgui_select_active_connection("con[0]");
sapgui_select_active_session("ses[0]");
/*Before running script, enter password in place of asterisks in logon function*/
sapgui_logon("JUNIOR{UserNumber}",
    "ides",
    "800",
    "EN",
    BEGIN_OPTIONAL,
    "AdditionalInfo=sapgui102",
    END_OPTIONAL);
```

Replaying SAP GUI Optional Windows

When working with SAP GUI Vuser Scripts, you may encounter optional windows in the SAP GUI client—windows that were present during recording, but do not exist during replay. If you try to replay your recorded script as is, it will fail when it attempts to find the missing windows.

VuGen's optional window mechanism performs the actions on a window only after verifying that it exists. The Vuser checks if the window indicated in the **Select active window** step exists. If the window is found during replay, it performs the actions as they were recorded in the script. If it does not exist, the Vuser ignores all window actions until the next **Select active window** step. Note that only SAP GUI steps (beginning with a **sapgui** prefix) are ignored.

To use this feature, in Tree view select the appropriate Select Active Window step and select **Run steps for window only if it exists** from the right-click menu.

To disable this feature and attempt to run these steps at all times, regardless of whether the Vuser finds the window or not, select **Always run steps for this window** from the right-click menu.

Configure the SAP Environment

This task describes configure and verify the SAP environment for use with VuGen.

VuGen support for the SAP GUI for Windows client, is based on SAP's Scripting API. This API allows Users to interact with the SAP GUI client, receive notifications, and perform operations.

The Scripting API is available only in recent versions of the SAP Kernel. In kernel versions that support scripting, the option is disabled by default. In order to use VuGen, first make sure that the SAP servers support the Scripting API, and enable the API on both the server and clients. For more information and to download patches, see the SAP OSS note #480149.

Use IPV6 with a SAP installation

Create an `SAP_IPV6_ACTIVE` environment variable and set the value to 1.

Check that SAPGUI Scripting API is enabled

Run the **VerifyScripting.exe** file from the **Additional Components\SAP_Tools\VerifySAPGUI** folder. For details, see ["Additional Components" on page 869](#). For more information, see the file **VerifyScripting.htm** provided with this utility.


Check the SAP GUI for Windows Client Patch Level

In your SAP GUI for Windows client, check the About box. The lowest patch level supported is version 6.20 patch 32.

Check the Patch Level

1. Open the SAP GUI logon window. Click the top left corner of the SAP Logon dialog box and select **About SAP Logon** from the menu.
2. The SAP version information dialog box opens. Verify that the Patch Level entry is 32 or higher.

Check the Kernel Patch Level

1. Log in to the SAP system.
2. Select **System > Status**.
3. Click the **Other kernel information**  button.
4. In the **Kernel Information** section, check the value of the **Sup. Pkg. lvl.**

The level must be greater than the level listed in the following chart depending on the SAP version you are using.

Software Component	SAP Release	Kernel Patch Level
SAP_APPL	31I	Kernel 3.1I level 650
SAP_APPL	40B	Kernel 4.0B level 903
SAP_APPL	45B	Kernel 4.5B level 753

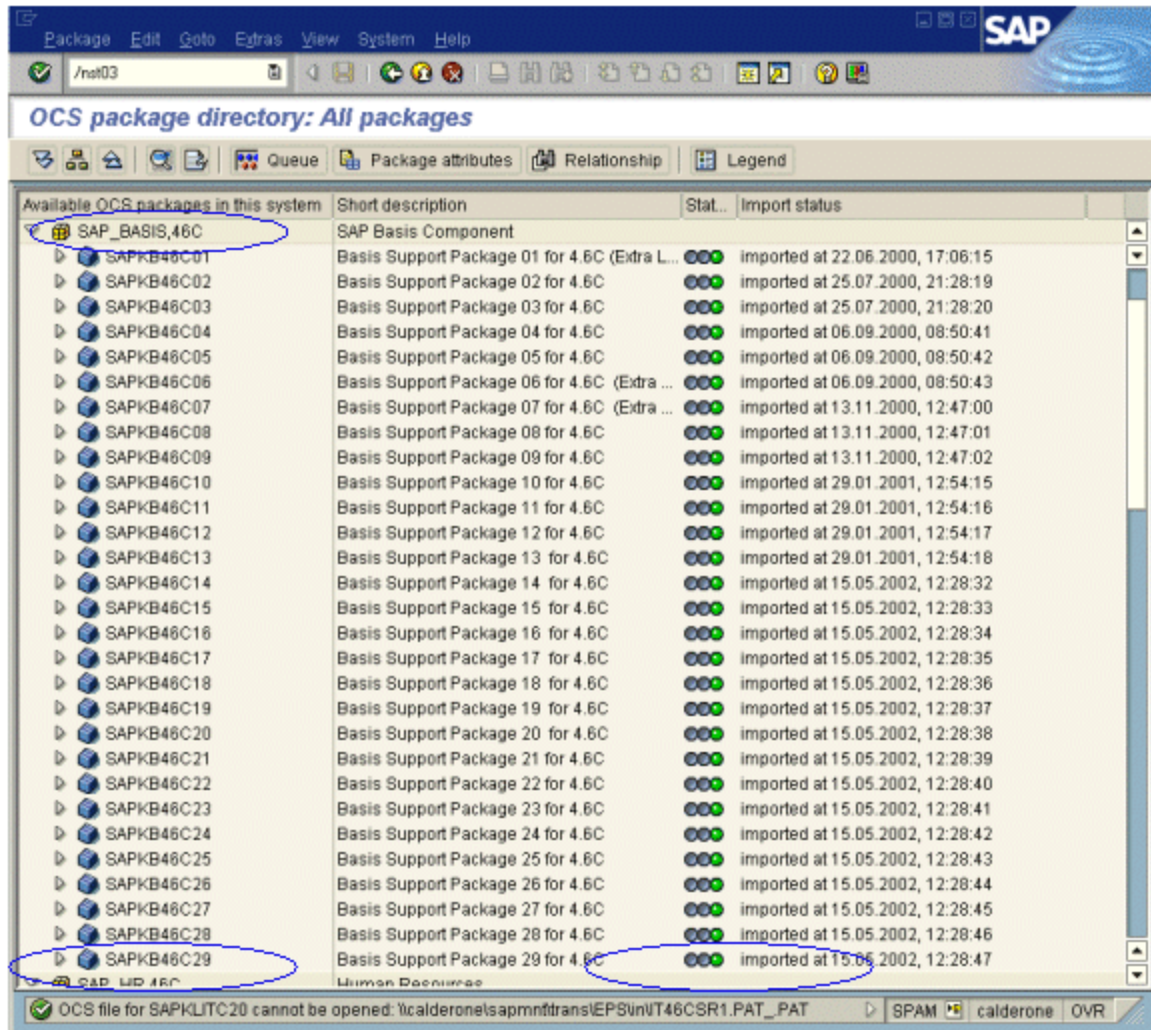
SAP_BASIS	46B	Kernel 4.6D level 948
SAP_BASIS	46C	Kernel 4.6D level 948
SAP_BASIS	46D	Kernel 4.6D level 948
SAP_BASIS	610	Kernel 6.10 level 360

Check the R/3 Support Packages

1. Log on to the SAP system and run the SPAM transaction.
2. In the **Directory** section, select **All Support Packages**, and click the **Display** button.
3. Verify that the correct package is installed for your version of SAP according to the table below.

Software Component	Release	Package Name
SAP_APPL	31I	SAPKH31I96
SAP_APPL	40B	SAPKH40B71
SAP_APPL	45B	SAPKH45B49
SAP_BASIS	46B	SAPKB46B37
SAP_BASIS	46C	SAPKB46C29
SAP_BASIS	46D	SAPKB46D17
SAP_BASIS	610	SAPKB61012

If the correct version is installed, a green circle appears in the Status column.

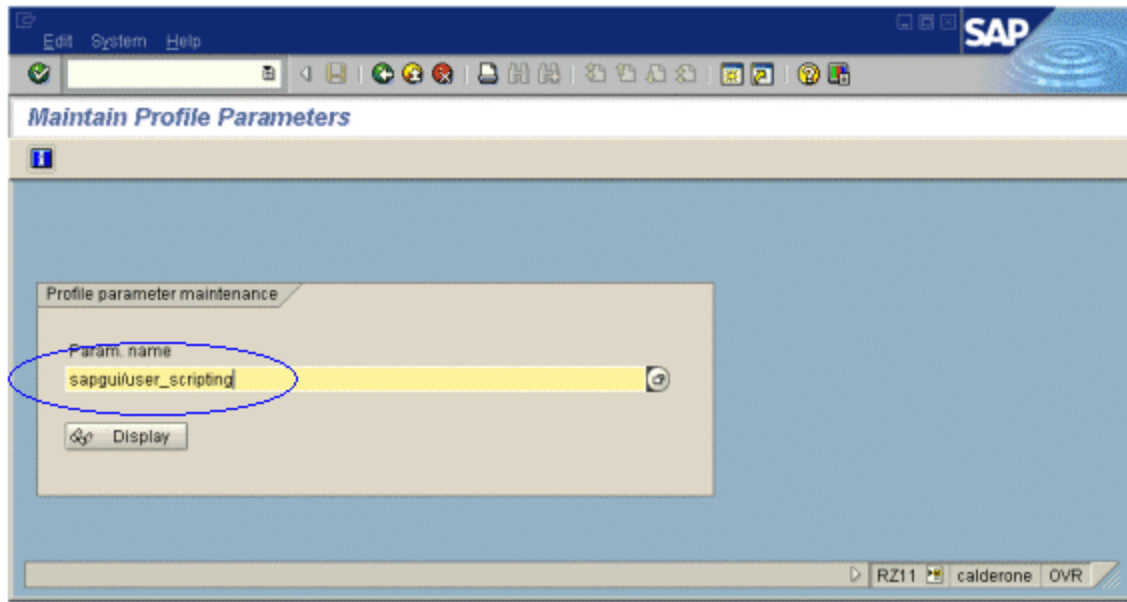


If you do not have the OCS package installed, download it from the www.sap.com Web site and install it. For more information, see the SAP OSS note #480149.

Enable scripting on the SAP Application Server

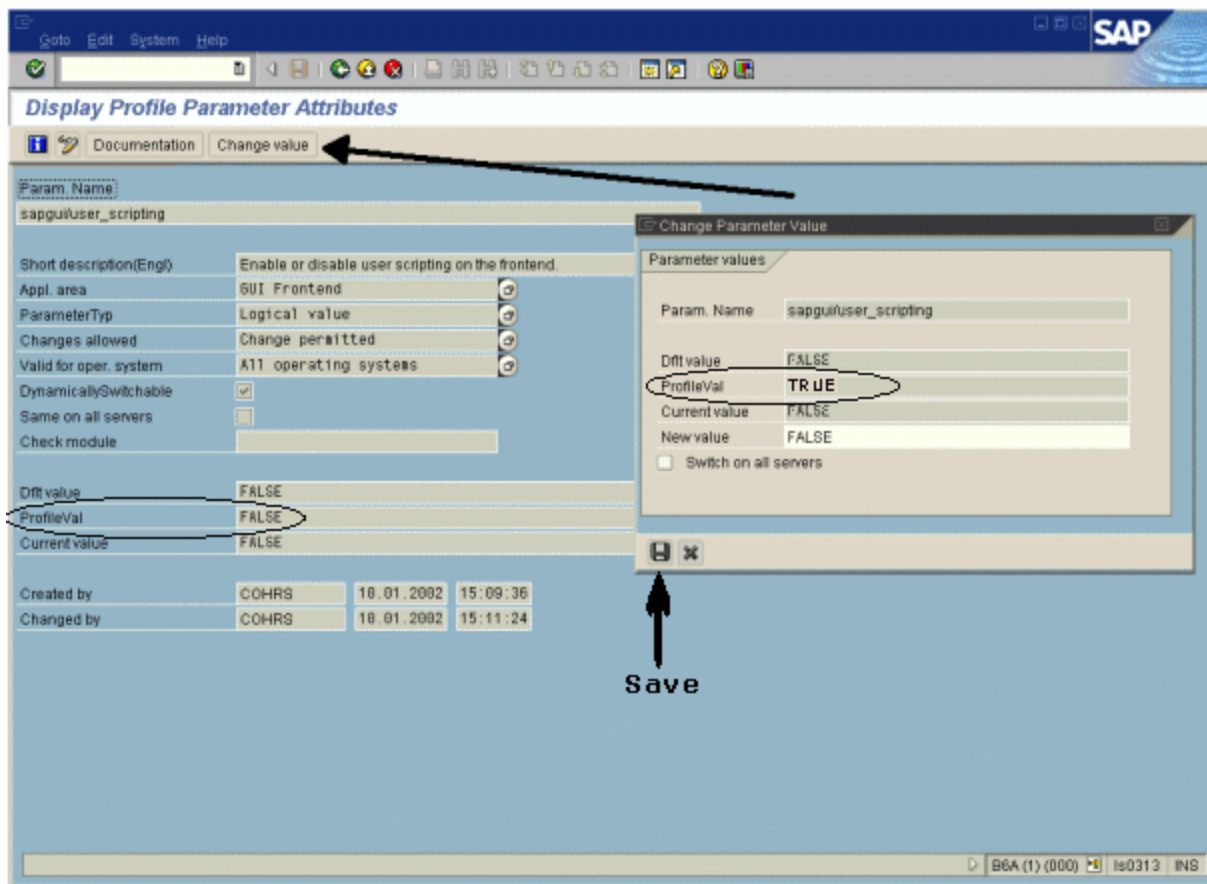
A user with administrative permissions enables scripting by setting the **sapgui/user_scripting** profile parameter to **TRUE** on the application server. To enable scripting for all users, set this parameter on all application servers. To enable scripting for a specific group of users, only set the parameter on application servers with the desired access restrictions. The following steps describe how to change the profile parameter.

1. Open transaction **rz11**. Specify the parameter name **sapgui/user_scripting** and click **Display**. The Display Profile Parameter Attributes window opens.



If **Parameter name is unknown** appears in the status bar, this indicates that you are missing the current Support Package. Import the Support Package that corresponds to the SAP BASIS and kernel versions of the application server, as described in the steps above.

2. If **Profile Val** is FALSE, you need to modify its value. Click the **Change value** button in the toolbar. The Change Parameter Value window opens. Enter TRUE in the **ProfileVal** box and click the **Save** button.



When you save the change, the window closes and **ProfileVal** is set to TRUE.

- Restart the application server to apply your changes.

If the updated **ProfileVal** did not change, even after restarting the server, then the kernel of the application server is outdated. Import the required kernel patch, as specified in the steps above.

Note: The Profile Value may be dynamically activated in the following kernel versions, using transaction rz11, without having to restart the application server.

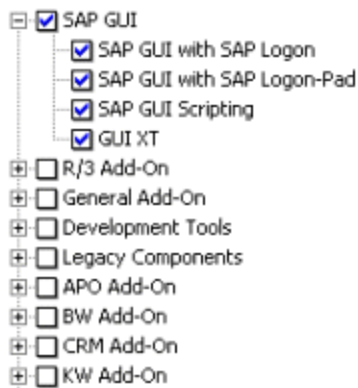
Release	Kernel Version	Patch Level
4.6B, 4.6C, 4.6D	4.6D	972
6.10	6.10	391
6.20	all versions	all levels

Enable scripting on SAP GUI 6.20 client

To allow VuGen to run scripts, you must also enable scripting on the SAP GUI client. You should also configure the client not to display certain messages, such as when a connection is established, or when a

script is attached to the GUI process. The following steps describe how to configure the SAP GUI client to work with .

1. **During installation.** While installing the SAP GUI client, enable the **SAP GUI Scripting** option.



2. **After installation.** Suppress warning messages. Open the Options dialog box in the SAP GUI client. Select the **Scripting** tab and clear the following options:

- **Notify when a script attaches to a running GUI**
- **Notify when a script opens a connection**

You can also prevent these messages from popping up by setting the values for **WarnOnAttach** and **WarnOnConnection** to 0 in the following registry key:

HKCU\SOFTWARE\SAP\SAPGUI Front\SAP Frontend Server\Security.

Examine the hierarchy of GUI Scripting objects - Optional

The **SAPGUI Spy** utility, is part of the "[Additional Components](#)" on [page 869](#). It examines the hierarchy of GUI Scripting objects, in open windows of **SAPGUI Client for Windows**.

To install the SAPGUI Spy component:

1. Copy the files **mscomctl.ocx**, **Msflxgrd.ocx** and **msvbvm60.dll** from the **SAP_Tools\SapGuiSpy\System32VBdls** folder to your Windows **System32** folder.
2. Register the files.
 - a. In the Windows **Run** box, enter: `regsvr32 <File name>`
 - b. Run the **SapSpy.exe** file from the **SAP_Tools\SapGuiSpy** folder.

Record SAP GUI Scripts

The following steps describe some prerequisites to recording a SAP GUI script.

Configure the application server for scripting

As a security precaution, scripting is disabled by default. In order to record, you need to enable scripting

on the application server. From the RZ11 transaction, set the following profile parameters as follows:

- sapgui/user_scripting TRUE
- sapgui/user_scripting_force_notification FALSE
- sapgui/user_scripting_set_readonly FALSE
- sapgui/user_scripting_disable_recording FALSE

Close SAPLogon application when recording with Multi

When recording a multi-protocol script in which the SAP GUI client contains Web controls, close the SAPLogon application before recording.

Use Modal dialog boxes for F1 Help

Instruct the SAP GUI client to open the F1 help in a modal dialog box as follows:

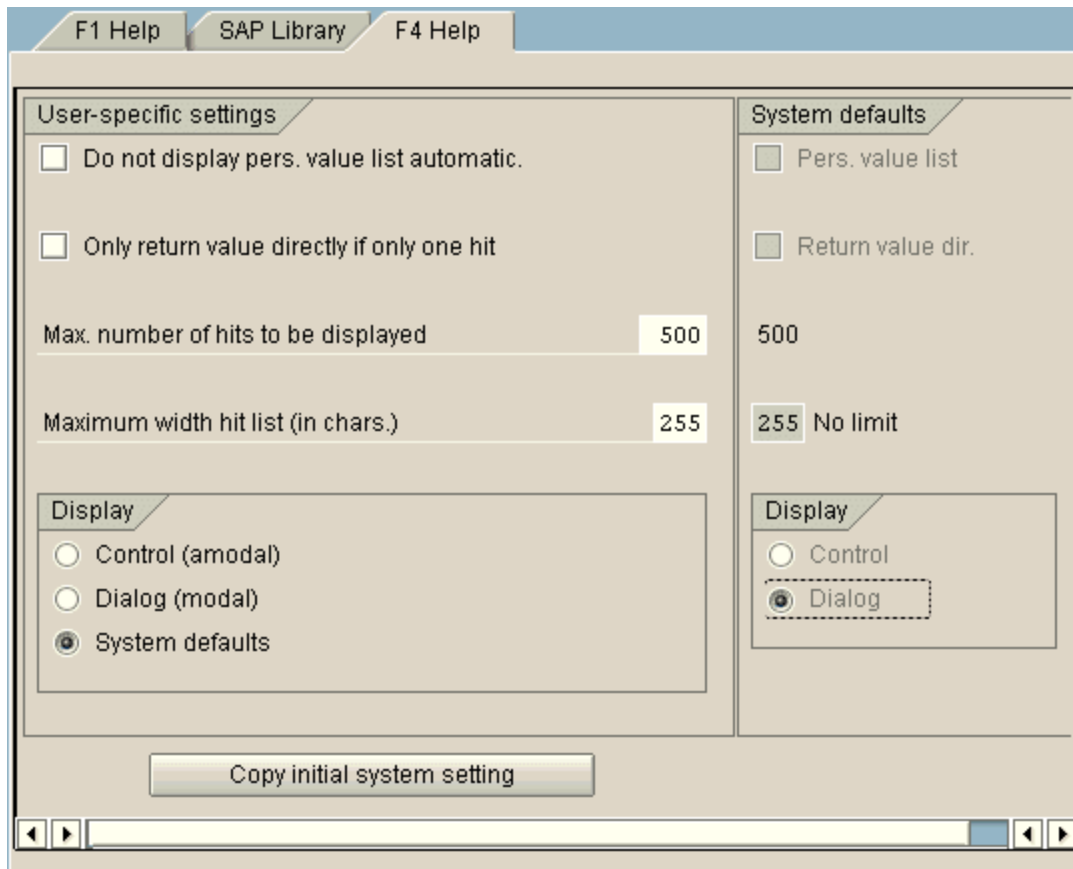
1. Select **Help > Settings**.
2. Click the **F1 Help** tab.
3. Select **in modal dialog box** in the Display section.

Use Modal dialog boxes for F4 Help

Note: This procedure can only be performed by the administrator.

Instruct the SAP GUI client to open the F4 help in a modal dialog box:

1. Make sure that all users have logged off from the server.
2. Select **Help > Settings**. Click the **F4 Help** tab.



3. In the Display section, select **System defaults**.
4. In the Display portion of the System defaults section, select **Dialog**.
5. Save the changes by clicking **Copy initial system setting**.
6. Verify that the status bar displays the message **Data was saved**.
7. Close the session and restart the service through the SAP Management Console.

Replay SAP GUI Scripts

The following steps describe prerequisites to replaying SAPGUI scripts.

Replace Masked Password

Replace the masked password in the **sapgui_logon** function generated during recording, with the real password. It is the second argument of the function, after the following user name

```
sapgui_logon("user", "pswd", "800", "EN");
```

For additional security, you can mask the password within the code. Right-click in the password text (the actual text, not *****) and select **Mask String**. VuGen inserts an **lr_unmask** function at the location of the password as follows:

```
sapgui_logon("user", lr_unmask("3ea037b758"), "800", "EN");
```

Display SAP GUI During Replay (optional)

When running a script for the first time, configure VuGen to show the SAP GUI user interface during replay, in order to see the operations being performed through the UI. Select **Replay > Runtime Settings > SAPGUI > General** node and select **Show SAP Client During Replay**. During a load scenario, disable this option, since it uses a large amount of system resources in displaying the UI for multiple Vusers.

Run SAP GUI Scripts from the Controller

The following steps describe tips for running SAP GUI scripts in a LoadRunner scenario.

LoadRunner Controller Settings

When working with a LoadRunner scenario, set the following values when running your script in a load test configuration:

- **Ramp-up.** One by one (to insure proper login) in the Scheduler.
- **Think time.** Random think time in the runtime settings.
- **Users per load generator.** 50 Vusers for machine with 512 MB of memory in the Load Generators dialog box.


Make Sure the Agent is Running in Process Mode

Make sure that the LoadRunner (or Performance Center) Remote Agent is running in Process mode. Service mode is not supported.

To check this, move your mouse over the agent's icon in the Windows task bar area, and read the description. If the description reads HPE Load Testing Agent Service, it is running as a service.



The following steps describe how to restart the agent as a process.

1.  Stop the agent. Right-click the LoadRunner Agent icon and select **Close**.
2. Run **magentproc.exe**, located in the **launch_service\bin** folder, under the LoadRunner installation.
3. To make sure that the correct Agent is launched the next time you start your machine, change the Start type of the Agent Service from Automatic to Manual. Then add a shortcut to **magentproc.exe** to the Windows Startup folder.
 - **Terminal Sessions.** Machines running SAP GUI Vusers may be limited in the number of Vusers that can run, due to the graphic resources available to that machine. To increase the number of Vusers per machine, open additional terminal server sessions on the Load Generator machines. Select **Agent Configuration** from **Start > All Programs > <product_name> > Advanced Settings**,

and select the **Enable Terminal Service** option. You specify the number of terminal sessions in the Load generator machine properties. For more information, see [Configuring Terminal Services Settings](#) in the HPE Controller User Guide.

Note: When the LoadRunner Agent is running in a terminal session, and the terminal session's window is minimized, no snapshots will be captured on errors.

Enhance SAP GUI Scripts

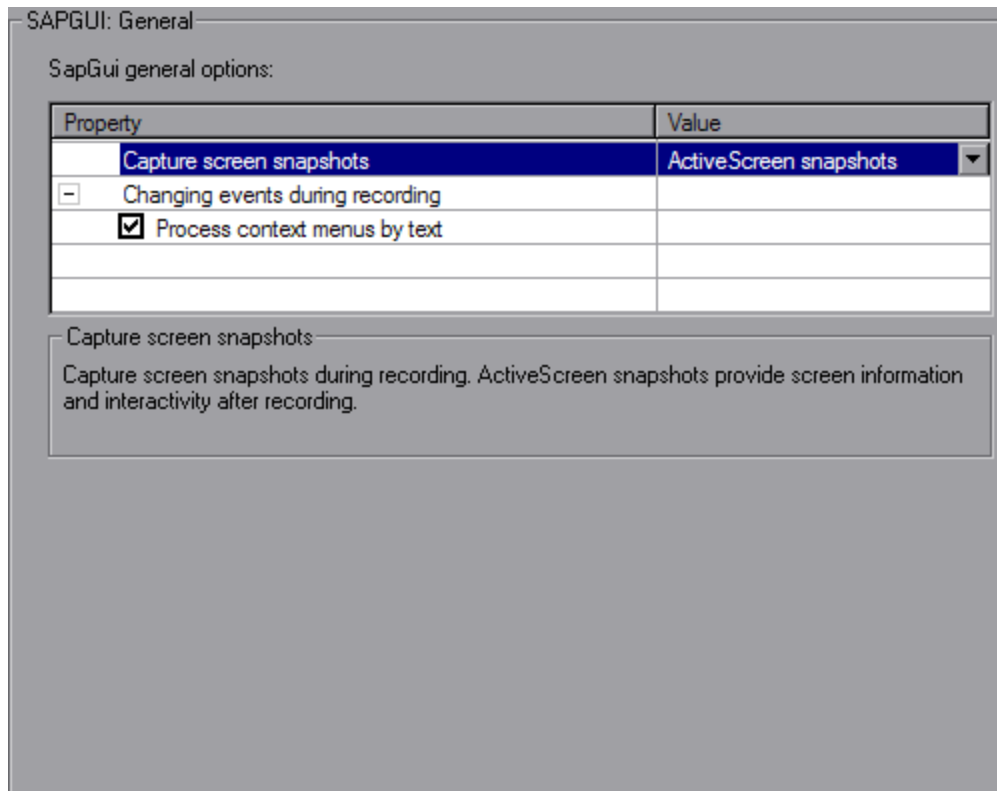
The following steps describe how to enhance SAP GUI protocol scripts.

Insert Steps Interactively into a SAP GUI Script

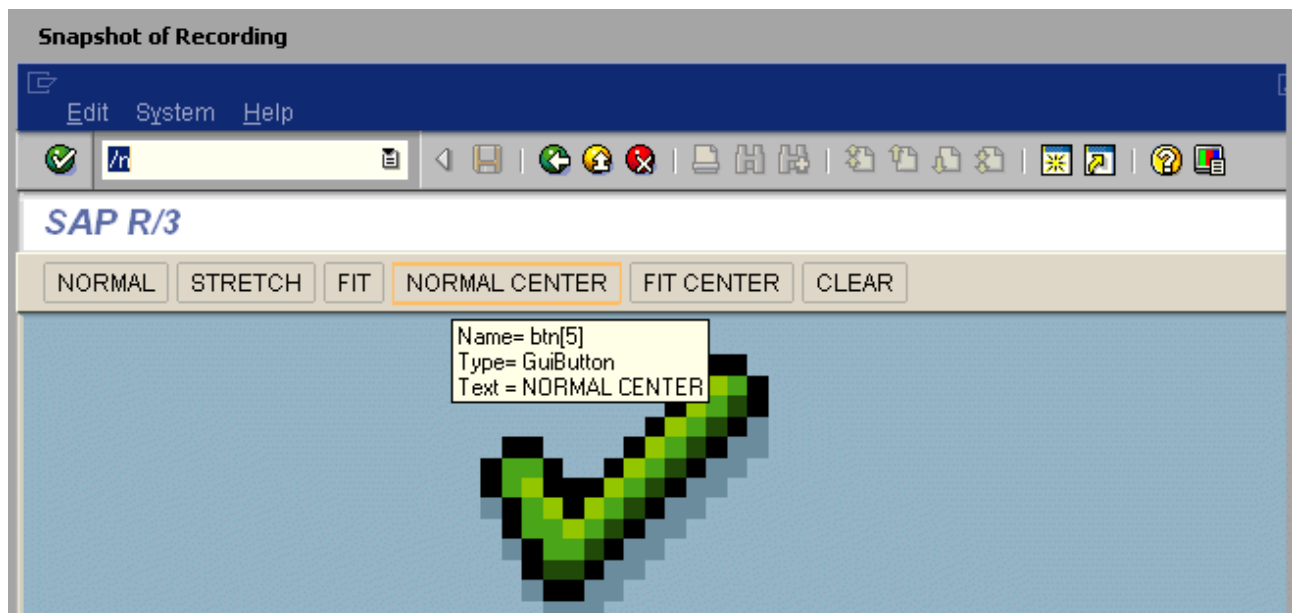
After recording, you can manually add steps to the script in either the **Editor** or **Step Navigator**. In addition to manually adding new functions, you can add new steps interactively for SAP GUI Vusers, directly from the snapshot. Using the right-click menu, you can add object-related steps.

When adding a step from within a snapshot, VuGen uses the Active Screen capability and determines the ID of each object in the SAP GUI client window (unless you disabled Active Screen snapshots in the ["SAPGUI > General Recording Options" on page 201](#)). The following steps describe how to insert a step interactively for a specific object.

1. Verify that you recorded the script when Active Screen snapshots were selected in the SAPGUI General node of the Recording Options (enabled by default).

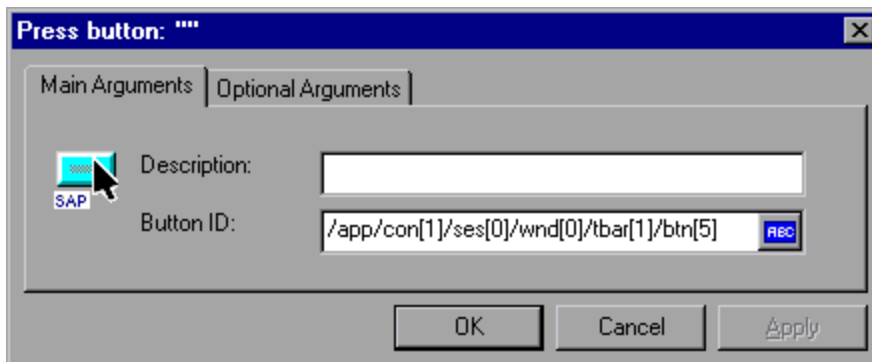


2. Click within the Snapshot pane.
3. Move the mouse over the object for which you want to add a function. Make sure that VuGen recognizes the object and encloses it with a box.



4. Right-click the object, click **Insert New Step**, and then select a step from the list of steps that are available for the object.

The step's Properties dialog box opens, with the Control ID of the object when relevant. For example, if you add a **Press Button** step, for the normal center button as shown above, the Properties box displays the following ID:



5. Enter a name for the object in the **Description** box. Click **OK**. VuGen inserts the new step after the selected step.

Note: You can get the Control ID of the object for the purpose of pasting it into a specific location. To do this, select **Copy Control ID** from the right-click menu. You can paste it into a Properties box or directly into the code from the Script view.

Add Verification Functions

When working with optional or dynamic windows or frames, you can use verification functions to determine if the window or object is available. An optional window is a window that does not consistently open during the SAP session. This function allow the Vuser script to continue running even if an optional window opens or an exception occurs.

The first example checks if a window is available. If the window is available, the Vuser closes it before continuing.

```
if (!sapgui_is_object_available("wnd[1]"))
    sapgui_call_method("{ButtonID}",
        "press",
        LAST,
        AdditionalInfo=info1011");
sapgui_press_button(....)
```

The next example illustrates a dynamic object in the ME51N transaction. The Document overview frame is optional, and can be opened/closed by the **Document overview on/off** button.

The code checks the text on the Document overview button. If the text on the button shows Document overview on, we click the button to close the Document overview frame.

```
if(sapgui_is_object_available("tbar[1]/btn[9]"))
{
```

```
sapgui_get_text("Document overview on/off button",
    "tbar[1]/btn[9]",
    "paramButtonText",
    LAST);
if(0 == strcmp("Document overview off", lr_eval_string("{paramButtonText}")))
    sapgui_press_button("Document overview off",
        "tbar[1]/btn[9]",
        BEGIN_OPTIONAL,
        "AdditionalInfo=sapgui1013",
        END_OPTIONAL);
}
```

Retrieve Information

When working with SAGUI Vusers, you can retrieve the current value of a SAP GUI object using the **sapgui_get_<xxx>** functions. You can use this value as input for another business process, or display it in the output log.

The following example illustrates how to save part of a status bar message in order to retrieve the order number.

1. Navigate to the point where you want to check the status bar text, and select **Insert New Step**. Select the **sapgui_status_bar_get_type** function. This verifies that the Vuser can successfully retrieve text from the status bar.
2. Insert an **if** statement that checks if the previous statement succeeded. If so, save the value of the argument using **sapgui_status_bar_get_param**.
This **sapgui_status_bar_get_param** function saves the order number into a user-defined parameter. In this case, the order number is the second index of the status bar string.

```
sapgui_press_button("Save (Ctrl+S)",
    "tbar[0]/btn[11]",
    BEGIN_OPTIONAL,
    "AdditionalInfo=sapgui1038",
    END_OPTIONAL);
sapgui_status_bar_get_type("Status");
if(0==strcmp(lr_eval_string("{Status}"),"Success"))
    sapgui_status_bar_get_param("2", " Order_Number ");
```

During test execution, the Execution log indicates the value and parameter name:

```
Action.c(240): Pressed button " Save (Ctrl+S)"
Action.c(248): The type of the status bar is "Success"
Action.c(251): The value of parameter 2 in the status bar is "33232"
```

Save Date Information

When creating scripts that use dates, your script may not run properly. For example, if you record the

script on June 2, and replay it on June 3, the date fields will be incorrect. Therefore, you need to save the date to a parameter during text execution, and use the stored value as input for other date fields. To save the current date or time during script execution, use the **lr_save_datetime** function. Insert this function before the function requiring the date information. Note that the format of the date is specific to your locale. Use the relevant format within the **lr_save_datetime** function. For example, for month.day.year, specify "%m.%d.%Y".

In the following example, **lr_save_datetime** saves the current date. The **sapgui_set_text** function uses this value to set the delivery date for two days later.

```
lr_save_datetime("%d.%m.%Y", DATE_NOW + (2 * ONE_DAY),  
                "paramDateTodayPlus2");  
sapgui_set_text("Req. deliv.date",  
                "{paramDateTodayPlus2}",  
                "usr/ctxtRV45A-KETDAT",  
                BEGIN_OPTIONAL,  
                "AdditionalInfo=sapgui1025",  
                END_OPTIONAL);
```

Additional SAP Resources

For more information, see the SAP website at www.sap.com or one of the following locations:

- **SAP Notes**- <https://websmp103.sap-ag.de/notes>
Note #480149: New profile parameter for user scripting on the front end
Note #587202: Drag =; Drop is a known limitation of the SAP GUI interface
- **SAP Patches** - <https://websmp104.sap-ag.de/patches>
SAP GUI for Windows - SAP GUI 6.20 Patch (the lowest allowed level is 32)

Troubleshooting and Limitations for SAP

This section describes troubleshooting and limitations for SAP GUI and SAP-Web protocols.

General SAPGUI limitations

- For LoadRunner users: If a business process in SAP GUI 7.30 includes selecting an item from a Combo-list, the business process may not run properly in LoadRunner versions 9.52 and higher.
Workaround: Add the path of the SAP GUI installation folder to the Windows PATH environment variable.
- Recording is not supported for the **SAP GUI Security** dialog box.
- Recording is not supported for standard Windows dialog boxes (for example Save or Open) which are opened from the SAP GUI client.
- If you encounter a warning during SAP GUI recording: "Sizing conflicts exist on the screen..." it may affect replay.

Workaround: Disable the warning in the SAP GUI application:

- a. Click the right-most button on the SAP GUI toolbar (or click Alt+F2) to open the Customize Local Layout screen.
- b. Select the **SAP Internal** sub-menu.
- c. Clear the **Enable dialog box for screen size** check box.

I was able to record a script, but why does replay fail?

In LoadRunner, make sure that the LoadRunner Remote Agent is running in Process mode. Service mode is not supported. For more information, see ["Run SAP GUI Scripts from the Controller" on page 649](#).

Why were certain SAP GUI controls not recorded?

Some SAP GUI controls are supported only in their menu or toolbar contexts. Try performing the problematic task using a different means—through a menu option, context menu, toolbar, and so on.

Why can't I record or replay any scripts in VuGen?

1. Verify that you have the latest patch of SAP GUI 6.20 installed. The lowest allowed patch level is patch 32.
2. Make sure that scripting is enabled. See ["Configure the SAP Environment" on page 640](#).
3. Verify that notifications are disabled in the SAP GUI for Windows client. Click the Customizing of Local Layout button or press ALT+F12. Click **Options** and select the Scripting tab. Clear both **Notify** options.

What is the meaning of the error popup messages that are issued when I try to run the script?

Certain SAP applications store the last layout for each user (such as which frames are visible or hidden). If the stored layout was changed since the script was recorded, this may cause replay problems. For Example, in the ME52N transaction, the **Document overview Off/On** button will change the number of visible frames.

If this occurs:

1. Navigate the transaction to the same point as it was during recording, before starting replay. You can use the Snapshot viewer to see the layout in which it was recorded.
2. Add statements to the script that bring the transaction to the desired layout during replay. For example, if an optional frame interferes with your replay, insert a verification function that checks if the frame is open. If it is open, click a button to close it. For verification examples, see ["Enhance SAP GUI Scripts" on page 650](#).

Can I use the single sign-on mechanism when running a script on a remote machine?

No, VuGen does not support the single sign-on connection mechanism. In your SAP GUI client, open the Advanced Options and clear the **Enable Secure Network Communication** feature. Note that you must re-record the script after you modify the Connection preferences.

Can VuGen record all SAP objects?

Recording is not available for objects not supported by SAP GUI Scripting. See your recording log for information about those objects.

Are all business processes supported?

VuGen does not support business processes that invoke Microsoft Office module controls, nor those that require the use of GuiXT. You can disable **GuiXT** from the SAP GUI for Windows client Options menu.

When I go to the Auto Logon node of the Recording Options, why is the list of server names empty?

This sometimes occurs when using SAP GUI Client 7.20. To resolve this issue, copy the **saplogon.ini** file from **%APPDATA%\SAP\Common** where **%APPDATA%** stands for the environment variable specifying the Application Data folder located directly below the user profile folder. Paste the file to the **%WINDIR%** folder (C:\Windows).



See also:

- For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

Siebel Web Protocol

Siebel Web Protocol Overview

The Siebel-Web protocol is similar to the standard Web Vuser protocol, with several changes in the default settings to allow it to work with the Siebel Customer Relationship Management (CRM) application.

You record typical activities in your Siebel session. VuGen records the actions and generates functions with a **web_** prefix, that emulate your actions.

You can also generate a Siebel Web Vuser script by analyzing an existing network traffic file (capture file). This method may be useful for creating Vuser scripts that emulate activity on mobile applications. For details, see ["Create a Vuser Script by Analyzing a Captured Traffic File" on page 685](#).

Siebel Web Recording Options and Runtime Settings

Before recording a Siebel Web Vuser script, set the following recording options:

- **Recording** node: **HTML-based script**

HTML Advanced - Script type: **A script containing explicit URLs only**

HTML Advanced - Non HTML-generated elements: **Do not record**

- **Advanced** node: Clear the **Reset context for each action** check box.

Before running a Siebel Web Vuser script, set the following runtime setting:

In the Runtime Settings, clear the **Simulate a new user on each iteration** check box in the **Browser Emulation** node.

Record Transaction Breakdown Information

VuGen provides a diagnostic tool for understanding the transaction components in your test—**transaction breakdown**. Using transaction breakdown, you can determine where the bottlenecks are and the issues that need to be resolved.

When preparing your script for transaction breakdown, we recommend that you add think time steps at the end of each transaction using the ratio of one second per hour of testing. For more information about adding think time steps, see ["Insert Steps into a Script" on page 340](#).

In order to record the transaction breakdown information, you need to modify your the parameterization functions in your script.

Prepare Your Script for Transaction Breakdown

1. Identify the script parameterization replacement of the Session ID.

```
/* Registering parameter(s) from source task id 15
// {Siebel_sn_body4} = "28eMu9uzkn.YGFFevN1FdrCfCC0c8c_"
// */
web_reg_save_param("Siebel_sn_body4",
    "LB/IC=_sn=",
    "RB/IC==;",
    "Ord=1",
    "Search=Body",
    "RelFrameId=1",
    LAST);
```

2. Mark the next **web_submit_data** function as a transaction by enclosing it with **lr_start_transaction** and **lr_end_transaction** functions.
3. Before the end of the transactions, add a call to **lr_transaction_instance_add_info**, where the first parameter, 0 is mandatory and the session ID has a SSQBD prefix.

```
lr_start_transaction("LoginSQLSync");
```

```
web_submit_data("start.swe_2",  
    "Action=http://design/callcenter_enu/start.swe",  
    "Method=POST",  
    "RecContentType=text/html",  
    "Referer=http://design/callcenter_enu/start.swe",  
    "Snapshot=t2.inf",  
    "Mode=HTML",  
    ITEMDATA,  
    "Name=SWEUserName", "Value=wrun", ENDITEM,  
    "Name=SWEPassword", "Value=wrun", ENDITEM,  
    "Name=SWERememberUser", "Value=Yes", ENDITEM,  
    "Name=SWENeedContext", "Value=false", ENDITEM,  
    "Name=SWEFo", "Value=SWEntryForm", ENDITEM,  
    "Name=SWETS", "Value={SiebelTimeStamp}", ENDITEM,  
    "Name=SWECmd", "Value=ExecuteLogin", ENDITEM,  
    "Name=SWEBID", "Value=-1", ENDITEM,  
    "Name=SWEC", "Value=0", ENDITEM,  
    LAST);  
lr_transaction_instance_add_info(0,lr_eval_string("SSQLBD:{Siebel_sn_body4}"));  
lr_end_transaction("LoginSQLSync", LR_AUTO);
```

Note: To avoid session ID conflicts, make sure that the Users log off from the database at the end of each session.

Siebel Web - Troubleshooting and Limitations

This section describes troubleshooting and limitations for Siebel Web Vuser scripts.



Tip: For general VuGen troubleshooting and limitations, see ["Troubleshooting and Limitations for VuGen" on page 868](#).

Recording High Interactivity Client

The Siebel High Interactivity client is only supported with a 32-bit Internet Explorer, version 9 and earlier. To record this type of session, check your browser version and downgrade if necessary. Alternatively, you can use proxy recording. For details, see ["Recording via a Proxy - Overview" on page 214](#).

Back or Refresh Error

An error message relating to **Back or Refresh** typically has the following text:

We are unable to process your request. This is most likely because you used the browser back or refresh button to get to this point.

Cause: The possible causes of this problem may be:

- The SWEC was not correlated correctly for the current request.
- The SWETS was not correlated correctly for the current request.
- The request was submitted twice to the Siebel server without the SWEC being updated.
- A previous request should have opened a frame for the browser to download. This frame was not created on the server probably because the SWEMethod has changed since the recording.

Same Values

A typical Web page response to the **Same Values** error is:

```
@0`0`3`3`0`UC`1`Status`Error`SWEC`10`0`1`Errors`0`2`0`Level0`0`ErrMsg`The same values for 'Name' already exist. If you would like to enter a new record, please make sure that the field values are unique.`ErrCode`28591`
```

Cause: The possible cause of this problem may be that one of the values in the request (in the above example, the value of the Name field) duplicates a value in another row of the database table. This value needs to be replaced with a unique value to be used for each iteration per user. The recommended solution is to replace the row ID with its parameter instead insuring that it will be unique.

No Content HTTP Response

A typical HTTP response for a **No Content HTTP Response** type error is:

HTTP/1.1 204 No Content

Server: Microsoft-IIS/5.0

Date: Fri, 31 Jan 2003 21:52:30 GMT

Content-Language: en

Cache-Control: no-cache

Cause: The possible causes of this problem may be that the row ID is not correlated at all or that it is correlated incorrectly.

Restoring the Context

The typical Web page response to the **Restoring the Context** type error is:

```
@0`0`3`3`0`UC`1`Status`Error`SWEC`9`0`1`Errors`0`2`0`Level0`0`ErrMsg`An error happened during restoring the context for requested location`ErrCode`27631`
```

Cause: The possible causes of this problem may be that the rowid is not correlated or that it is correlated incorrectly.

Cannot Locate Record

The typical Web page response to the **Cannot locate record** type error is:

@0`0`3`3`0`UC`1`Status`Error`SWEC`23`0`2`Errors`0`2`0`Level0`0`ErrMsg`Cannot locate record within view: Contact Detail - Opportunities View applet: Opportunity List Applet.`ErrCode`27573`

Cause: The possible causes of this problem may be that the input name SWERowId does not contain a row ID for a record on the Web page. This input name should have been parameterized. The parameter's source value may have changed its location.

End of File

The typical Web page response to the **End of File** type error is:

@0`0`3`3`0`UC`1`Status`Error`SWEC`28`0`1`Errors`0`2`0`Level0`0`ErrMsg`An end of file error has occurred. Please continue or ask your systems administrator to check your application configuration if the problem persists.`ErrCode`28601`

Cause: The possible causes of this problem may be that the input name SWERowId does not contain a row ID for a record on the Web page. This input name should have been parameterized. The parameter's source value may have changed its location.


Unable to Retrieve Search Categories

The typical Web page response to the **Unable to Retrieve Search Categories** type error is:

Cause: A possible cause of this problem may be that the search frame was not downloaded from the server. This occurs when the previous request did not ask the server to create the search frame correctly.


Silverlight Protocol

Silverlight Protocol - Overview

 **Caution:** This protocol is supported for replay only. Support for this protocol will be discontinued in future versions.

Microsoft Silverlight is a web application framework that supports graphics, animations, and interactivity. The Silverlight protocol includes the Web - HTTP/HTML protocol as a subset, as well as a number of additional functions and runtime settings.

Silverlight - Troubleshooting and Limitations

 **Caution:** This protocol is supported for replay only. Support for this protocol will be discontinued in future versions.

This section describes troubleshooting and limitations for the Silverlight protocol.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

- REST services do not generate Silverlight service calls. However they can be replayed.
- Duplex (Polling) Binding for WCF Web Services is not supported.
- Silverlight 4 and 5 clients are supported, however applications developed using the new communication features such as net.tcp binding are not supported.
- The VuGen snapshot viewer does not support Silverlight controls.
- The Silverlight Protocol does not support applications which use Japanese, Korean, Simplified Chinese, and Traditional Chinese.

Teradici PCoIP (PC over IP) Protocol

The PCoIP (Teradici PC over IP) protocol supports testing on the Teradici Pervasive Computing Platform and VMware HorizonView.

Because only images are passed from the server, the user can perform **connect**, **disconnect**, **mouse click**, and **key press** actions only.

PCoIP client

The PCoIP client, is shipped with LoadRunner (resides in <LoadRunner installation>\bin\pcoip). The PCoIP client supports VMWare HorizonView.

Alternatively, you can download and install the official Teradici client software from Teradici (<http://www.teradici.com/product-finder/client-downloads>).

Record a PCoIP test

1. Click the **Start Recording** button.
2. In the Start Recording dialog, select **Windows Application** under **Recording mode** and enter the PCoIP client path name in the **Application** field. For example,

```
<LoadRunner installation>\bin\pcoipo\pcoip_client.exe
```

3. Enter other information as appropriate to your test and click **Start Recording**.
4. Follow the directions on the Teradici client screens to connect to the server.
5. Record your business process from within the client.



Note: A call to **Copy image to the clipboard** from the snapshot context menu may throw an **OpenClipboard Failed** exception. In that event, close all open PCoIP clients.

Enhance the recorded script

During recording, strings (alphanumeric keyboard presses) are recorded until some other type of activity occurs, such as mouse clicks or non-alphanumeric key presses.

When typing is slow, VuGen may break up a typing session into a number of invocations of **pcoip_type()** interleaved with **lr_think_time()**. By manually editing the script, these calls can be concatenated. This will facilitate parameterization if you need the string to be substituted with a parameter.

Sample Script

This script enters "Hello World" in an open application.

```
Action()
{

    pcoip_set_auth("alex-qa", "mydomain", lr_unmask("56ec1f82347be574b867f74a72"));

    pcoip_connect("MYD-SERVER", "123.456.789.abc", "MYD-SERVER", "4172", 1);

    pcoip_set_display(980, 556);

    pcoip_key("NUM_LOCK_KEY", 0);

    pcoip_mouse_click(18, 537, LEFT_BUTTON, 0, "snapshot3");

    pcoip_key("ENTER_KEY", 0);

    pcoip_key("h", MODIF_SHIFT);

    pcoip_type("ello", 0);

    pcoip_key("w", MODIF_SHIFT);

    pcoip_type("orlx", 0);

    pcoip_key("BACKSPACE_KEY", 0);

    pcoip_key("d", 0);

    pcoip_key("ENTER_KEY", 0);

    pcoip_key("TAB_KEY", 0);

    pcoip_disconnect();

    return 0;
}
```

TruClient Protocol

TruClient is a tool for recording Web-based applications. The TruClient engine records your actions as you navigate through your business process. It creates a script in real-time, allowing you to see the steps as they are performed in a sidebar.

For details, see the [TruClient Help Center](#) (select the relevant version).

If you are working locally, you can open the TruClient Help Center from the **Start** menu — go to **HPE Software > Documentation**.

Convert TruClient scripts to TruClient - Web

Scripts recorded in TruClient version 12.02 and earlier can be replayed only in the browser on which they were recorded. You can convert these scripts to TruClient - Web so that they can be interactively replayed on any supported browser. After the converted scripts have been interactively replayed on a particular browser, the scripts can be used on that browser in load tests.

Note: TruClient script conversion to Web - HTTP/HTML is not supported for converting to a JavaScript language script.

Converting multiple scripts

You can convert multiple scripts either from VuGen or from the TruClient Launcher.



Tip: You can also use this process for a single script if the script is contained in a parent folder that contains no other scripts.

1. To run the batch conversion tool, select **Tools > TruClient batch conversion tool**. The TruClient Scripts Converter dialog box is displayed.
2. In the **Source folder** field, define the path to the folder that contains the scripts to be converted. All of the scripts under the source folder must be single-browser scripts that can be converted by the tool.
3. The **Destination folder** define the path where the converted scripts will be saved.

Converting a single script

VuGen provides an option to convert a single script. To convert a single script:

1. Open the script in VuGen.
2. Select **Tools > Convert Script to TruClient Web** to display the conversion dialog.
3. Define a name and location for the converted script.
4. Click Convert. The converted script is displayed in the Solution Explorer.

Web - HTTP/HTML Protocol

The Web - HTTP/HTML Vuser protocol is one of VuGen's *Web Vuser* and *Mobile* protocols. This section includes information that is specific to the Web - HTTP/HTML Vuser protocol. For information that is generic to all Web Vuser protocols, see ["Web Protocols \(Generic\)" on page 691](#).

Web - HTTP/HTML Protocol - Overview



Note: This topic applies to Web - HTTP/HTML Vuser scripts only.

The Web - HTTP/HTML Vuser protocol emulates communication between a browser and Web server on an HTTP or HTML level.

When should you use the Web - HTTP/HTML Vuser protocol?

You can use the Web - HTTP/HTML Vuser protocol for browser applications that include applets and VB script, and for non-browser applications.

Use the Web - HTTP/HTML Vuser protocol when the client and the server communication is done over http/s communication, and the complexity of the communication does not require content modification. If content modification is required, consider using the TruClient protocol. For further information about TruClient, see the [TruClient Help Center](#) (select the relevant version).

Web - HTTP/HTML Vuser Technology

You use VuGen to develop Web - HTTP/HTML Vuser scripts. To record a Web - HTTP/HTML Vuser script, you navigate through a web site - performing typical user activities. VuGen records your actions and generates a Web - HTTP/HTML Vuser script. The script contains detailed information about the recorded traffic. When you run the script, the resulting Vuser emulates a user accessing the Internet.

For details, see ["Web Vuser Technology" on page 692](#).

Learn how to develop a Web - HTTP/HTML Vuser script

The table below displays a list of the LoadRunner documentation that relates to the process of developing a Web - HTTP/HTML Vuser script.

Topic	Description
Creating a Web (HTTP/HTML) Vuser script	See VuGen's generic documentation about creating Vuser scripts ["Creating Vuser Scripts - Overview" on page 113].

Topic	Description
Recording	<p>In addition to the generic documentation about recording Vuser scripts ["Recording - Overview" on page 131], see:</p> <ul style="list-style-type: none"> • "Recording Levels - Overview" on page 211 • Recording Options: <ul style="list-style-type: none"> • "General > Script Recording Options" on page 169 • "General > Protocol Recording Options" on page 166 • "General > Recording - Recording Options" on page 166 • "Network > Mapping and Filtering Recording Options" on page 185 • "HTTP Properties > Advanced Recording Options" on page 175 • "Correlations > Rules Recording Options" on page 151 • "Correlations > Configuration Recording Options" on page 149 • "Data Format Extension > Chain Configuration Recording Options" on page 157 • "Data Format Extension > Code Generation Recording Options" on page 161
Correlating	<p>In addition to the generic VuGen documentation on correlating Vuser scripts ["Correlation Overview" on page 232], see:</p> <ul style="list-style-type: none"> • "Web Manual Correlation in Code" on page 272 • "Data Format Extensions (DFEs) - Overview" on page 704 • "Using the VuGen JavaScript Engine" on page 669
Replaying	<p>In addition to the generic VuGen documentation about replaying Vuser scripts ["Replay Overview" on page 277], see:</p> <ul style="list-style-type: none"> • "Browser Emulation - Overview" on page 698 • "Working with Cache Data" on page 702 • Runtime Settings <ul style="list-style-type: none"> • General - Run Logic, Pacing, Log, Think Time, Additional Attributes, Miscellaneous • Network > Speed Simulation Node • Browser > Browser Emulation Node • Internet Protocol > Proxy Node • Internet Protocol > Preferences Node • Internet Protocol > Download Filters Node • Internet Protocol > ContentCheck Node

Topic	Description
Debugging	See VuGen's generic documentation about debugging Vuser scripts ["Debugging Overview" on page 313].
Parameterizing	In addition to the generic VuGen documentation on parameterizing Vuser scripts ["Parameterization Overview" on page 342], see: <ul style="list-style-type: none"> • "Data Format Extensions (DFEs) - Overview" on page 704 • "Using the VuGen JavaScript Engine" on page 669
Adding Load Testing functionality	In addition to the generic VuGen documentation on adding load testing functionality ["Enhancing a Script for Load Testing - Overview" on page 321], see: <ul style="list-style-type: none"> • "Text and Image Verification (Web Vuser Scripts) - Overview" on page 694
Viewing Test Results	See VuGen's generic documentation about viewing test results ["Replay Summary Pane" on page 108].
Misc	The following miscellaneous topics are applicable to Web - HTTP/HTML Vuser scripts: <ul style="list-style-type: none"> • "Create Web - HTTP/HTML scripts from TruClient Scripts " on page 686 • "JavaScript Web HTTP Scripts" below • "Convert a Web - HTTP/HTML Vuser Script into a Java Vuser Script" on page 675 • "Web Snapshots - Overview" on page 697 • "Record HTTP/2 " on page 683 • "Record Applications Using Smooth Streaming" on page 687 • "Create a PCAP File" on page 831

See also:

- ["Web Vuser Types" on page 693](#) for a full list of Web Vuser protocols

JavaScript Web HTTP Scripts

You can record Web HTTP/HTML Vuser scripts in JavaScript. You can also regenerate a C language script to convert it to JavaScript. The JavaScript engine is Google's V8 engine and is ECMAScript 5 compatible.



Note: VuGen's code regeneration overwrites all manual changes that you made to a recorded script; it only regenerates the recorded functions.

JavaScript Vuser scripts support asynchronous behavior and functionality. For details, see ["Create an Asynchronous Vuser Script" on page 381](#).

JavaScript Vuser scripts support Virtual User Tables (VTS). For more information see ["VTS and Parameterization" on page 344](#)

To run a script faster, you can disable JavaScript debugging at **Runtime Settings -> Preference -> JavaScript**.

Recording your Vuser Script in JavaScript

When you create a Web HTTP/HTML Vuser script, you are notified that you can choose to generate the script in JavaScript. See ["General Options Tab" on page 86](#) to disable the message.

To record your script in JavaScript:

1. Click **Record > Recording Options**. Select **Script** in the General menu on the left.
2. Select **JavaScript** in the Scripting Language dropdown list.



3. Click OK.

After recording the script, the default C files are replaced by JS files.


Note: Once you have recorded your script the Language option in the **Recording Options** dialog box is disabled and cannot be changed. You can change the language of a script after it is recorded only by regenerating it.

Converting a C Script to JavaScript

You convert a C script to JavaScript by regenerating a recorded script. You cannot convert a script that you coded.

Caution: If you changed a recorded script, your changes are lost when you regenerate.

To regenerate C scripts into JavaScript:

1. Click  or **Record > Regenerate Script**. A warning appears that any changes to the code are overwritten.
2. Click **Options**. The Regenerate Options dialog box appears.

3. In the Script section, select the target language to convert the script to.
4. Click **OK** in the Regenerate Options dialog box. Click **OK** to approve regenerating the code.

JavaScript Function Libraries

The native LoadRunner JavaScript library includes string functions, database connectivity functions and XML functions among others. Native library functions are available from four global objects:

- web: Web/HTTP Protocol API.
- lr: LoadRunner native API.
- vtc: VTC multiple connections API.
- lrvtc: VTC single connections API.

Custom Libraries

In addition to LoadRunner standard functions, you can create JavaScript function libraries, or use JavaScript Language libraries such as underscore, lodash, moment, and so on.

Before you can invoke custom functions, add your custom library into the Extra files section. Your custom library must have a 'js' extension.

To activate auto-completion on your library, right click on the js file name and select **Add to Parsing List**.

Note: If you regenerate your script in another language, any additional files, such as a user-defined .js file, are removed from the parsing list.

External Libraries

In addition, the **lr.require** and **lr.loadLibrary** functions enable you to import JavaScript files at runtime. Third-party libraries are supported for libraries that are not MS-Windows or document-object based. Therefore, asynchronous functions and libraries, such as promise libraries, are not supported.

lr.loadLibrary is used to “include” a single file into the script. The code is evaluated and its definitions are added to the global scope of the runtime context. For example: `lr.loadLibrary('external.js');`

lr.loadLibrary can be used for browser based libraries.

lr.require loads a CommonJS file or NPM module into a string, then wraps it with a function and evaluates it. Additional required files are loaded recursively along the “require” link.



Example: `var lodash = lr.require("../node_modules\\lodash");`

`var sum = lodash.sum([1,2,3,4]);`

See also:

- ["Debugging" on page 313](#)
- The JavaScript section of the Function Reference (**Help > Function Reference**)

Using the VuGen JavaScript Engine

Applies to: Web - HTTP/HTML Vuser scripts written in C only

What is the VuGen JavaScript Engine?

Typically, Web - HTTP/HTML Vuser scripts contain C code. The built-in JavaScript Engine enables you to insert snippets of JavaScript code into the C code.

What can I do with JavaScript in a Vuser script?

You can insert JavaScript code into a Web - HTTP/HTML Vuser script to manipulate text strings that are included in the request and response messages that are sent between the client and server. Manipulating strings is often useful for correlation and parameterization purposes. Typical string manipulations include converting decimal to hexadecimal, encoding and decoding Base64, URL encoding and decoding, and accessing object values inside JSON-formatted data.

Note: It is possible to perform many of these string manipulation procedures by using the built-in DFEs (Data Format Extensions). For details, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Inserting JavaScript code into a Vuser script may also be useful when client-side logic is implemented in JavaScript. Inserting snippets of the original client-side JavaScript code into the Vuser script removes the requirement of having to re-write the JavaScript logic into C code to be included in the Vuser script.

You can use JavaScript code in a Vuser script to execute an **XMLHttpRequest**. This allows you to generate and send HTTP or HTTPS requests using standard Javascript APIs. Such APIs include, for example, sending asynchronous requests, assigning callbacks to handle responses, reading responses in XML format. An **XMLHttpRequest** used this way may replace a call to an action step such as **web_url** or **web_custom_request**.

Why use JavaScript snippets?

Although it may be possible to achieve the required functionality by using C code alone, including JavaScript in a Vuser script may be beneficial for the following reasons:

- JavaScript often offers a more intuitive, easier to implement solution than C.
- The JavaScript regular expression library simplifies the challenge of working with regular expressions.
- There are numerous JavaScript libraries that assist with string manipulation.
- Client-side logic is often implemented in JavaScript. Inserting snippets of the original JavaScript code removes the requirement of having to translate the JavaScript client logic into C code.

Can I use the JavaScript Engine in Vuser scripts of all protocols?

No, the JavaScript Engine enables you to insert JavaScript into Web - HTTP/HTML Vuser scripts only.

What are some scenarios in which the JavaScript Engine may be useful?

Including JavaScript code in a Vuser script may be useful in the following scenarios:

Scenario 1: Converting a decimal number to its hexadecimal representation

In this scenario, the response that a Vuser sends to the server must include a 13-digit timestamp in hexadecimal format. For example, the date/time stamp "1234567891234" must be converted by the Vuser into hex and sent as "11F71FB0922". LoadRunner does not include any standard functionality to perform this conversion, and developing the required C code is not trivial. This problem can be resolved by inserting the following JavaScript code into the Vuser script:

```
web_js_run(
    "Code=getHexTimestamp();",
    "ResultParam=HexTS",
    SOURCES,
    "Code=getHexTimestamp=function(){return new Date().getTime().toString(16).toUpperCase();}", ENDITEM,
    LAST);

lr_output_message("[%s]",lr_eval_string("{HexTS}"));
```

Scenario 2: Base64 encoding and decoding

The request and response messages that are sent between the client and server include data that is encoded using a Base64 coding scheme. Because the data is encoded, it is often difficult or impossible to parameterize or correlate the raw data. The data must be decoded before it can be parameterized or correlated, and then re-encoded before being sent to the server. By including JavaScript snippets in a Vuser script, you can access external JavaScript libraries that implement the required Base64 decoding and encoding functionality.

Note: It is possible to implement Base64 decoding and encoding using the built-in Base64 DFE (Data Format Extension). For details, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Scenario 3: URL encoding and decoding

In this scenario, the request and response messages that are sent between the client and server include URLs that are encoded using JavaScript URL encoding. By including a JavaScript snippet in a Vuser script, you can access the JavaScript **encodeURIComponent** and **decodeURIComponent** functions that perform the required decoding and encoding procedures.

Note: It is possible to implement URL decoding and encoding using the built-in URL Encoding DFE (Data Format Extension). For details, see ["Data Format Extensions \(DFEs\) - Overview" on](#)

page 704.

Scenario 4: Accessing objects inside JSON-formatted data

In this scenario, the request and response messages that are sent between the client and server include data in JSON format. To access objects inside the JSON formatted data, you can include a JavaScript snippet inside the Vuser script to access the JavaScript **eval()** function.

Note: It is possible to access objects inside JSON-formatted data by using the built-in JSON-To-XML DFE (Data Format Extension). For details, see ["Data Format Extensions \(DFEs\) - Overview" on page 704.](#)

Scenario 5: Using XMLHttpRequest

You can use JavaScript code in a Vuser script to execute an **XMLHttpRequest** to download stock quotes from a specified site. For an example of how to execute an **XMLHttpRequest**, see ["JavaScript Engine: XMLHttpRequest Example" on page 674.](#)

Scenario 6: Pre-existing client-side JavaScript code

To access a particular Web site, the Vuser must submit a user name and an encrypted password. The server sends a server hash to the browser to enable the browser to generate the required encrypted password. The code to generate the hash is complicated, and exists in JavaScript. This JavaScript code can be included in the Vuser script, removing the requirement to re-write the JavaScript logic into C code.

What are the API functions that I can use in a Vuser script to execute Javascript code?

The following API functions are available for including JavaScript in a Vuser script:

1. **web_js_run:** Runs the specified JavaScript code.
2. **web_js_reset:** Clears the JavaScript context.

You use the **web_js_run** function to include JavaScript code in a Vuser script. Using the **web_js_run** function, you can either insert the required JavaScript code into the Vuser script, or you can reference a file that contains the required JavaScript code.

Example of inserted JavaScript code

The following is an example of how to include JavaScript code directly from the Vuser script:

```
web_js_run(  
    "Code=xor((LR.getParam('buffer'), 0xFFFF));",  
    "ResultParam=param",  
    LAST);
```

Example of a referenced file that contains JavaScript code

The following is an example of how to include JavaScript code by calling a file that contains the JavaScript code:

```
web_js_run(
    "File=XMLHttpRequest_sync_sample.js;",
    "ResultParam=param",
    LAST);
```

For details on the above functions, and examples of how they can be used, see the [Function Reference \(Help > Function Reference\)](#).

Can I use JavaScript to access any "internal" API functions?

JavaScript in a Vuser script gives you access to a number of "internal" API functions that can be called directly from a **web_js_run** function in the JavaScript code. These functions are used primarily for managing parameters, but also enable you to log specified messages, record data, and run **XMLHttpRequest**.

Javascript-specific API functions

API Function	Description	Arguments
LR.advanceParam (parameter)	Advances the specified parameter to the next value in the file.	parameter. The name of the parameter to advance. Must be a parameter of type file or unique number.
LR.setParam (name, value)	Saves a string to a parameter, creating the parameter if it does not exist.	name. The name of the parameter in which to save the value. value. The value.
LR.freeParam (name)	Deletes a dynamic parameter at runtime, freeing its buffer.	name. The parameter name.
LR.getParam (name)	Returns the value of the specified parameter.	name. The parameter name.

API Function	Description	Arguments
LR.log(text, level)	Logs a message.	text. The message. level. One of the following: <ul style="list-style-type: none"> • "Error" • "Warning" • "Standard" • "Extended" • "Status" example: LR.log("text", "Error");
LR.userDataPoint(name, value)	Records a user-defined data point for analysis.	name. The name of the data point. Do not begin a data-point name with any of these strings: HTTP, NON_HTTP, RETRY, mic_, stream_, mms_ value. The numeric value.

How do I enable the Javascript engine for Vusers?

To run JavaScript from within a Vuser script, you must enable the JavaScript engine for the Vuser script. To enable the JavaScript engine, open the **Replay > Runtime Settings > Internet Protocol > Preferences** view. Go to the **JavaScript** section and select the **Enable running JavaScript code** option.



Note: Enabling this option causes the creation of a JavaScript Runtime Engine, even if there are no JavaScript steps in the script.

How do I configure the JavaScript engine?

You use the Vuser script's runtime settings to configure the product's custom JavaScript engine.

To access the JavaScript Engine (JSE) runtime settings, select **Replay > Runtime Settings > Internet Protocol > Preferences**, and expand the **JavaScript** section.

- **JavaScript Engine runtime size:** Specifies the size of the allocated JavaScript Engine Runtime memory, in kilobytes. This value may need to be increased when running a large number of Vusers.
- **JavaScript Engine stack size per-thread:** Specifies the size of each Vuser thread in the JavaScript Engine memory, in kilobytes. This value may need to be increased for large objects or deep stack calls.

For user interface details, see "[Preferences View - Internet Protocol](#)" on page 291.

What is the connection between the VuGen's JavaScript Engine and VuGen's JavaScript Protocol?

There is no connection between VuGen's JavaScript Engine and VuGen's JavaScript Protocol.

Troubleshooting

If you encounter difficulties when implementing JavaScript Engine support, review the items below for possible solutions.

1. Make sure that VuGen's JavaScript Engine is enabled. For details, see [How do I enable the JavaScript Engine?](#)
2. Javascript limits you to adding up to 9,000 operands in one function. For example, if you are combining strings, "str1"+"str2"+"str3"+..."str9000", you can only add up to 9,000 strings.
3. Memory issues
 - If the **Simulate a new user on each iteration > Clear cash on each iteration** runtime setting is selected, **web_js_reset** is called automatically at the start of each iteration.
 - If **Simulate a new user on each iteration > Clear cash on each iteration** is not set, avoid excessive memory consumption by inserting **web_js_reset** calls in your Vuser script at points where you no longer need the saved context.

For details on the **web_js_reset** function, see the Function Reference (**Help > Function Reference**).

For details on the runtime settings, see the hints below the option in the runtime settings view.

4. Performance issues

If you are experiencing performance issues, modify the **JavaScript** runtime settings. For details, see ["Preferences View - Internet Protocol" on page 291](#).

JavaScript Engine: XMLHttpRequest Example

Note: This topic applies to Web - HTTP/HTML Vuser scripts written in C only.

VuGen's JavaScript Engine enables you to include JavaScript code in a Vuser script. For details on the JavaScript Engine, see ["Using the VuGen JavaScript Engine" on page 669](#).

The example below shows how you can use a JavaScript **XMLHttpRequest** object in a Web (HTTP/HTML) Vuser script. In this example, the **XMLHttpRequest** object enables the Vuser to download a stock quote from finance.example.com, and then to save the value to a parameter for future use.

The script section below shows a **web_js_run** function that has been inserted into a Vuser script. The **web_js_run** function includes a reference to a file called XMLHttpRequest_sync_sample.js. This file contains the JavaScript code that executes the **XMLHttpRequest** function.

```
web_js_run(  
    "Code=getQuotes(LR.getParam('symbol'))";  
    "ResultParam=param";  
    SOURCES,  
    "File=XMLHttpRequest_sync_sample.js", ENDITEM,  
    LAST);
```

The contents of the XMLHttpRequest_sync_sample.js file are shown below.

```
var req2;

function getQuotes(mySymbol)
{
var myURL="http://download.finance.example.com/d/quotes.csv?s="+mySymbol+"&f=1&e=.csv";

    req2 = false;
    // branch for native XMLHttpRequest object
    try {
        req2 = new XMLHttpRequest();
    } catch(e) {
        req2 = false;
    }
    if(req2) {
        req2.open("GET", myURL, false);
        req2.send("");
    }
    return 1*req2.responseText;
}
```

- For additional examples of code used with the JavaScript Engine, see the **Function Reference (Help > Function Reference)**.
- For information about the **XMLHttpRequest** object, see http://www.w3schools.com/ajax/ajax_xmlhttprequest_send.asp.

Convert a Web - HTTP/HTML Vuser Script into a Java Vuser Script

Note: This topic applies to Web - HTTP/HTML and Java Vuser scripts only.

VuGen provides a utility that enables you to convert a Web - HTTP/HTML Vuser script into a Java Vuser script. This also allows you to create a hybrid Vuser script for both Web and Java.

1. Create an empty **Java Vuser** script and save it.
2. Create an empty **Web (HTML/HTTP)** Vuser script and save it.
3. Record a session into the Web (HTML/HTTP) Vuser script.
4. Replay the Web (HTML/HTTP) Vuser script. When it replays correctly, cut and paste the entire script into a text editor and save it as a text file (.txt).
In the text file, modify any parameter braces from the Web type, "{ }" to the Java type, "< >".
5. Open a DOS command window and go to the <Installation_folder>/**dat** folder.
6. Type the following command:

```
<Installation Folder>\bin\sed -f web_to_java.sed filename > outputfilename
```

where **filename** is the full path and filename of the text file you saved earlier, and **outputfilename** is the full path and filename of the output file.

7. Open the output file, and copy its contents into your Java Vuser script action section at the desired location.

If you are pasting the contents into an empty custom Java template (Java Vuser type), modify the line containing `public int action()` as follows:

```
public int action() throws Throwable
```

This change is done automatically for recorded Java users (RMI and CORBA).

8. Parameterize and correlate the Vuser script as you would with an ordinary Java script, and run the script.

Create a Script for a REST API

In recent years, Representational State Transfer (REST) has become a popular model for software architecture, especially for Web-based applications. Today, most large websites such as Twitter, Google, Flickr, and so forth, use REST APIs. Using the **web_rest** function, you can create a load test script for a REST API.

REST API components

A REST API call consists of the following components:

- **Uniform Resource Identifier (URI).** A string comprised of the host, the path of the functional component, and the Query string, with key-value pairs. For example :

```
http://www.shopping.hpe.com/en_US/home-office/-  
/products/Tablets/Tablets?SearchParameter=ElitePad
```

- **Method.** The action to perform, such as GET, POST, PUT, and DELETE.
- **Data.** The data to send to the server, usually in JSON format.

All of these components are included as parameters of a **web_rest** function. For details, see the Function Reference (**Help > Function Reference**).

Using the REST Step Properties dialog box

The REST Step Properties dialog box helps you to generate **web_rest** functions. To open the REST Step Properties dialog box, select **Design > Insert in Script > REST API**, or right-click in the Script Editor and select **Insert > REST API**. The dialog box includes various tabs that help you to generate and check the required **web_rest** functions.

Tab	Function
Build Step	<p>Helps you to define the web-rest step:</p> <ul style="list-style-type: none"> Specify the required URL and select a method. The request body editor area changes slightly depending on whether the selected method can have a body attached to it or not. Specify the required URL parameter key-value pairs. <p>Note: As you add key-value pairs, VuGen adds them to the URL above.</p> <p>Click +Add to add additional key-value pairs.</p> <ul style="list-style-type: none"> Specify the required parameters for the headers. You can enter any string as the header name, or you can select one of the common headers from the drop down list. Click +Add to add additional header parameters. Specify the required values for the body parameters. Click +Add to add additional body parameters.
Preview	Shows a preview of the request code that is generated, based on the data that is contained in the Build Step tab. This is the code that is sent to the server when the resulting web_rest function is run.
Step Results	<p>Shows the HTTP response that was returned by the server after the most recent step run. The response is divided between the Body, Cookies, and Headers tabs.</p> <p>Note: The status of the response appears in the top right corner of the dialog box.</p>

- After you have entered the required attributes for the **web_rest** function, you can click **Run Step** to run the step to test that it functions correctly. The HTTP response to the step run is displayed in the Step Results tab. The HTTP status of the test run appears in the top right corner of the dialog box.
- After you have entered and checked the required attributes of the step, click **Insert Step** to insert the **web_rest** function into your script, at the cursor location.

Note: After you click **Insert Step** to insert the step into your script, VuGen adds the step to the step history list. Click **Step History** [located on the left of the REST Step Properties dialog box] to show the step history list. You can select any step in the list to insert the step details into the Build Step tab. You can then modify the step, run it, and insert it into the script. A separate step history list is maintained for each computer on which VuGen is installed. If required, click **Clear History** to clear the step history list.



Tip: To open the REST Step Properties dialog box to edit an existing **web_rest** function, right-click inside the function and then select **Show Arguments**.

Create script steps for a Web - HTTP/HTML script that calls a REST API

1. Create a Web (HTTP/HTML) Vuser script. For details see, ["Creating Vuser Scripts - Overview" on page 113](#).
2. Record the REST API application while you perform typical business processes in the application. After you finish recording, add new **web_rest** calls using the following format, where the attributes are **URL**, **Method**, **ResType**, **Body**, and **BodyFilePath**:

```
web_rest("<request_name>",  
        "<Attribute_List>",  
        LAST);
```

3. (Optional) To reference a file with the JSON data instead of entering the actual text:
 - a. In the Solution Explorer, right-click the **Extra Files** node and select **Add Files to Script** to add the .json data file.
 - b. Replace the **Body** argument with **BodyFilePath=<file_name.json>**.
 - c. Allow JSON files. Select **Tools > Options > Scripting > Script Management** and add .json to the Allowed Extensions list.

Examples

The following example shows a REST API function that updates values using a **PUT** action:

```
web_rest("update customer info",  
        "URL=http://myServer/customers/{cust_id}",  
        "Method=PUT",  
        "ResType=JSON",  
        "Body={ \"address\": \"yyyy\", ... }",  
        LAST);
```

The next example shows a REST API function that uses data from a file when updating values using a **POST** action:

```
web_rest("create new customer",  
        "URL=http://myServer/customers/",  
        "Method=POST",  
        "ResType=JSON",  
        "BodyFilePath = c:\\my_data\\data.json" /* BodyFile for large content */  
        LAST);
```

Using Emulation to Record Mobile Applications

Using an emulator to test mobile applications often provides a robust solution for many mobile devices. To test mobile devices using emulation, you install a third party emulation application on your local computer, and then record events using the "Windows Application" method in VuGen's Start Recording dialog box. (For details, see ["Start Recording Dialog Box" on page 221.](#)) After selecting the "Windows Application" method, the Start Recording dialog box requires the following emulator settings in order to start the emulator:

- **Emulator to record:** Enter this in the **Application** field.
- **Command line:** Enter this in the **Program arguments** field.
- **Working directory:** Enter this in the **Working directory** field.

The table below shows emulator download sites and examples of emulator settings for various mobile operating systems:

Link to Download SDK	Emulator to record (example)	Command line (example)	Working directory (example)
Android			
Android Emulator Download	C:\Program Files\Android\android-sdk-windows\tools\emulator.exe	@Android_v2.2	C:\Program Files\Android\android-sdk-windows\tools
Blackberry			
Blackberry Emulator Download	C:\Program Files\Research In Motion\BlackBerry Smartphone Simulators 6.0.0\6.0.0.337 (9800)\fledge.exe	/app=Jvm.dll /handheld=9800 /session=9800 /app-param=DisableRegistration /app-param=JvmAlxConfigFile:9800.xml /data-port=0x4d44 /data-port=0x4d4e /pin=0x2100000A	C:\Program Files\Research In Motion\BlackBerry Smartphone Simulators 6.0.0\60.0.337 (9800)
Windows 8			
Windows Mobile Emulator Download	C:\Program Files\Microsoft SDKs\Windows Phone\v7.0\Tools\XDE Launcher\XdeLauncher.exe	"Windows Phone 7" "Windows Phone 7 Emulator"	C:\Program Files\Microsoft SDKs\Windows Phone\v7.0\Tools\XDE Launcher

Recording with a Google Android Emulator

Google Android Emulator Version 2.0 and above


If you experience problems when recording with Google Android Emulator Version 2.0 and above, apply the following workaround:

1. Enter a new Port Mapping by selecting **Recording Options > Network > Mapping and Filtering** and select **New Entry** in the **Port Mapping** section.
2. Specify a Target server and port.


3. Enter a second Port Mapping entry without changing any details.

Server Entry - Port Mapping


— Socket Service —

 Target Server : (Any Server)
Port : (Any)
Service ID : (Auto Detect)
Service Type : TCP
Record Type : Proxy
Connection Type : Plain


— SSL Configuration —

 SSL Version : SSL 2/3
SSL Ciphers : (Default OpenSSL Ciphers)

☐ Use specified client-side certificate (Base64/PEM)

 Client Cert :
Password :

☐ Use specified proxy-server certificate (Base64/PEM)

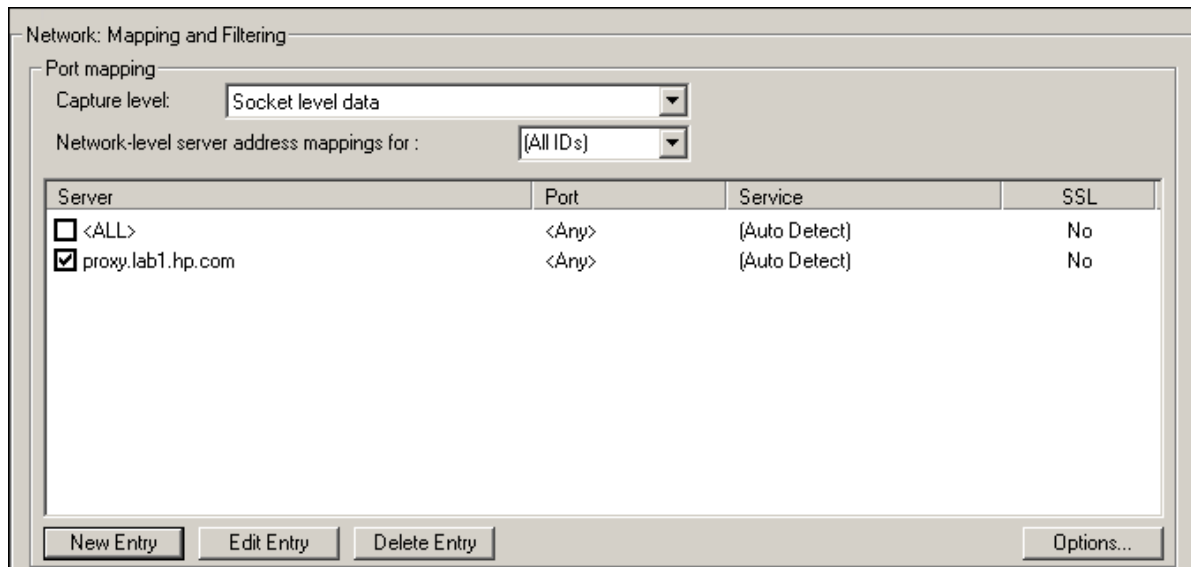
 Proxy Cert :
Password :

Test SSL

Description
Set the security level of connection. This entry aides the analyzer in decrypting secure connections.

Update Cancel

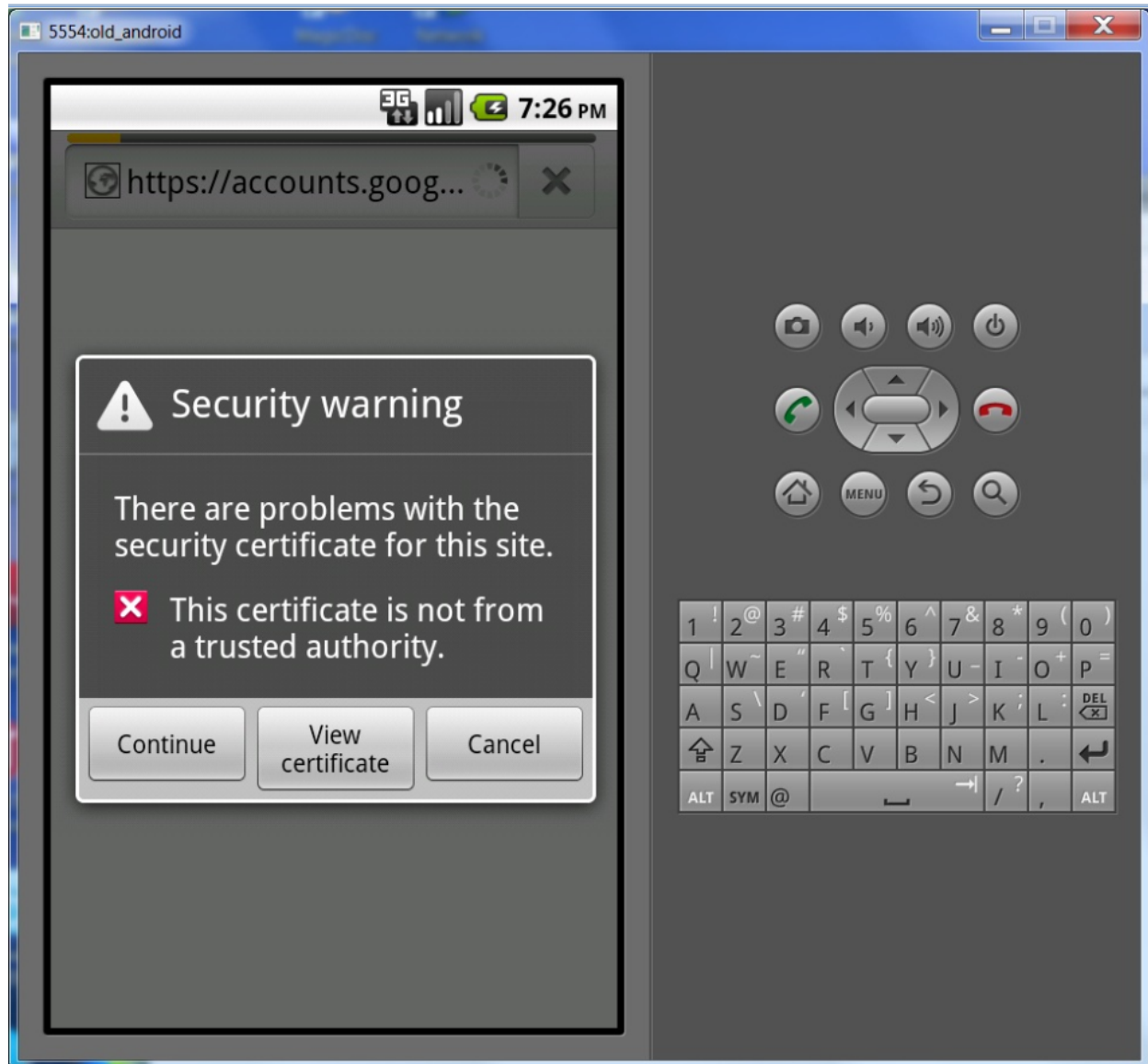
4. Disable the second entry so all traffic is handled by the first entry in the Port Mapping.



Google Android Emulator - Resolved limitation

The following limitation was resolved in Google Android Emulator Version 4.03:

While recording an SSL site, you may encounter a warning message stating that there are problems with the certificate for the site. Click **Continue** to proceed with the recording.



Record HTTP/2

HTTP/2 was standardized as a successor to HTTP 1.1 in May 2015. This version does not replace HTTP. Instead, it improves the way HTTP requests and responses are sent.

HTTP/2 allows multiplexing and concurrency, letting a client send multiple requests successively on the same TCP connection. This eliminates the need for multiple connections between the client and server.

In HTTP/2, the header size is substantially smaller. In addition, the server can push information to the client, even before it was requested.

All of these changes contribute to HTTP/2 being faster and more efficient than its predecessor.



Tip: To determine if the Web site you are testing supports HTTP/2, perform an Internet search



for an HTTP/2 detection tool. Most common browsers support HTTP/2 over TLS.

To record a Web - HTTP/HTML script that uses the HTTP/2:

1. Create a Web (HTTP/HTML) Vuser script. For details see, "[Creating Vuser Scripts - Overview](#)" on [page 113](#).
2. Select **Record > Recording Options > Network > Maps and Filtering > Options** to open the Advanced Port Mapping Settings dialog box.
3. In the **SSL Version** dropdown, select **TLS ALPN**. Click **Update**.
4. Start recording and perform typical business processes on your application.
5. To confirm that recording was done with HTTP/2, examine the headers that were recorded in the script.

To verify that HTTP/2 was used for replay, enable **Advanced Trace** in the **Log** runtime settings. After replay, see the Replay log and note the references to HTTP/2.

Record and Replay on servers with SNI Enabled

Server Name Indication (SNI) is an extension of the TLS protocol, in which a client indicates which hostname it is attempting to connect to at the start of the handshaking process. This allows multiple certificates to be sent from a single server. It is often used on cloud servers, such as AmazonCloud.

You can control whether VuGen sends an SNI extension to SSL handshakes using:

web_set_sockets_option("TLS_SNI", <value>);

Where:

Value	Description
"1"	Default. Enable extension and send the server name value derived from the URL (host name)
"0"	Disable extension. Do not send an SNI extension with SSL handshakes.
"<server name>"	Manually specify the SNI extension value for the next handshake only.

Additional SNI Guidelines

VuGen uses the following guidelines for SSL handshakes:

During recording:

- If SNI support is detected, no special step is added.
- If SNI is not detected, a
`web_set_sockets_option("TLS_SNI", "0");`
statement is added to the beginning of the script.

During replay:

- When the `<server_name>` option is used, the specified server name is used for the next handshake only.
Subsequent handshakes revert to the standard enabled (1) or disabled (0) behavior.
- If no `web_set_sockets_option("TLS_SNI", <value>);` statement exists, or if a `web_set_sockets_option("TLS_SNI", "1");` exists, the default SNI extension is sent with each SSL handshake.
Therefore, scripts created prior to support of this option will run with SNI enabled.
To prevent this behavior, add `web_set_sockets_option("TLS_SNI", "0");` to your script.

Create a Vuser Script by Analyzing a Captured Traffic File

Note: This task is applicable to the following Vuser types: Web - HTTP/HTML, Flex, SAP - Web, and Siebel - Web.

This topic describes how to use an existing network traffic file (capture file) to generate a Vuser script. This method may be useful for creating Vuser scripts that emulate activity on mobile applications.

Note: In addition to the procedure described below, you can create a Vuser script by right-clicking a captured traffic file (.pcap, .cap, .saz, or .har) in Windows Explorer, and then selecting **Create VuGen script**.

1. Create a new Vuser script in VuGen.
Note that this functionality is available only for those Vuser protocols listed in the note at the top of this topic.
2. Click **Start Recording** to open the Start Recording dialog box.
3. From the list of Recording methods, select **Captured Traffic File Analysis**.
4. Locate and select your capture file (**pcap**, **cap**, **saz** (Fiddler), or **har**).
For details about creating a new capture file in a Windows, Linux, or mobile environment, see ["Create a PCAP File" on page 831](#).
5. Specify the client side filter. This is the IP address of the *client* whose traffic you want to use to generate the Vuser script. VuGen typically detects the client side filter by analyzing the capture file.
6. You can use the recording options to specify server side filters by clicking **Recording Options > Mapping and Filtering**.
7. Add a list of the SSL attributes for the servers being analyzed. Use the **Add**, **Edit**, and **Remove** buttons to manage the entries in the list. The **Add** button opens the Add SSL Attribute dialog box, allowing you to add a server and specify its IP address, port, certificate file, and password if required.

Note: The SSL attribute list is visible only after you select a **pcap** capture file that contains SSL data and requires a certificate. This list is not available for **har** and **saz** files — instead,


configure the SSL through Fiddler.

8. Click **Start Recording**. VuGen analyzes the capture file, and generates a Vuser script.

Create Web - HTTP/HTML scripts from TruClient Scripts

Script development with TruClient is fast, but more Vusers can be run with VuGen Web - HTTP/HTML than with TruClient. You can combine the advantage of fast script development, and the advantage of running many Vusers, by developing your script with TruClient and then converting it to a Web - HTTP/HTML script.

Note: During the conversion, comments and APIs are added to the Web - HTTP/HTML protocol script that document the conversion process.

1. Create a TruClient script. See **Record a TruClient Web script** in the [TruClient Help Center](#) (select the relevant version).
2. Save the script and close the TruClient Side Bar.
3. From the VuGen toolbar, click the  **Convert** button to convert the script.
4. After the script is generated, review the script, keeping in mind that the Web - HTTP/HTML protocol records on the transport level. For example, you may need to address correlation or parameter issues in your converted script.

When you add a converted script to a scenario, VuGen offers to create a new Vuser group for the script, provided that the original TruClient script is in the specified path. Having a separate group enables you to view its results separately.

Watch a video: [TruClient to Web HTTP/HTML script conversion](#)

Limitations

Conversion of TruClient scripts to Web - HTTP/HTML scripts does not support converting steps that call 127.0.0.1 (localhost).

Record Streaming Media in Web - HTTP/HTML

The Web streaming functions emulate communication between a client and a server that provides streaming media using one of the streaming protocols, HTML5 or HLS.

If the application under test runs on mobile devices (browser or APPs), use proxy recording. If the application's client is a desktop browser, application or plugins, you can use either hook-based or proxy based recording.

Set timeouts, number of retries, and configure snapshots and logging in the Run-Time Settings under **Network > Streaming**. Make sure your timeouts do not conflict with the settings of **Internet Protocol > Preferences > HTTP**. Step time in streaming media includes the play time, so set the Internet Protocol timeouts to allow for playing.

The downloaded media is not decoded. Instead, it is consumed by a player emulator. You control the emulator with the streaming functions: wait, seek, play, and pause. Other streaming functions get and set attributes of the stream. This protocol does not record user actions on a media player widget. Except for open and close, all the functions are added manually.



See also:

- VuGen Function Reference. These functions are under **Web Vuser Functions > Alphabetical List of Web Vuser Functions**. The names all start with "web_stream_".

To open the Function Reference from a machine with LoadRunner installed, click **Start > All Programs > HPE Software > HPE LoadRunner > Documentation > Function Reference**. In icon-based desktops, such as Windows 8, search for **Function** and select **Function Reference** from the results.

- HTTP Streaming Monitoring
- Streaming State Statistics Graphs

Record Applications Using Smooth Streaming

Smooth Streaming is an Internet Information Services (IIS) Media Services extension that provides high-quality video streaming to clients over HTTP. Smooth Streaming adapts the stream rate and quality by monitoring the local bandwidth and video playback performance of the client while traditional streaming delivers the content at a fixed rate and quality.

To prepare a script for load testing for applications that use Smooth Streaming:

1. Create a Web (HTTP/HTML) Vuser script. For details see, ["Creating Vuser Scripts - Overview" on page 113](#).
2. Look for the "Manifest" request at the start of the streaming communication:

```
web_custom_request("Manifest",  
    "URL=http://mediadl.microsoft.com/mediadl/iisnet/smoothmedia/E  
xperience/BigBuckBunny_720p.ism/Manifest",  
    "Method=GET",  
    "Resource=0",  
    "RecContentType=text/xml",  
    "Referer=",  
    "Snapshot=t11.inf",  
    "Mode=HTTP",  
    LAST);
```

3. Following the "Manifest" request, you should find a number of streaming requests:

```
web_custom_request("Fragments(video=0)",
    "URL=http://mediadl.microsoft.com/mediadl/iisnet/smoothmedia/E
xperience/BigBuckBunny_720p.ism/QualityLevels(350000)/Fragments(video=
0)",
    "Method=GET",
    "Resource=1",
    "RecContentType=video/mp4",
    "Referer=",
    "Snapshot=t12.inf",
    LAST);

web_custom_request("Fragments(audio=0)",
    "URL=http://mediadl.microsoft.com/mediadl/iisnet/smoothmedia/E
xperience/BigBuckBunny_720p.ism/QualityLevels(64000)/Fragments(audio=0
)",
    "Method=GET",
    "Resource=1",
    "RecContentType=video/mp4",
    "Referer=",
    "Snapshot=t13.inf",
    LAST);
```

Create and configure parameters to emulate different bandwidths than the ones that were recorded. For example:

- a. In the streaming request, replace QualityLevel with a parameter named 'qualityLevel'.

```
web_custom_request("Fragments(video=0)",
    "URL=http://mediadl.microsoft.com/mediadl/iisnet/smoothmedia/E
xperience/BigBuckBunny_720p.ism/QualityLevels({qualityLevel})/Fragment
s(video=0)",
    "Method=GET",
    "Resource=1",
    "RecContentType=video/mp4",
    "Referer=",
    "Snapshot=t12.inf",
    LAST);
```

- b. Configure the 'qualitylevel' values that will be used during each iteration of the load test in the ["Parameter Properties Dialog Box" on page 359](#).
4. Replay the script and verify that the size of the response from each request corresponds to the value of the parameter that was sent.

Common Web Recording Problems

No events are being recorded

- **Problem:** The events counter on the recording toolbar keeps increasing, while the generated script is empty.
- **Possible Cause:** VuGen's recording mechanism may be unable to identify HTTP data.

- **Solution:** Ensure that your application indeed uses HTTP Web traffic.

If your application uses SSL connections, make sure you have the correct SSL version (SSL 2, SSL 3, TLS) in the Port Mapping dialog box (**Record > Recording Options > Network > Port Mapping > Options**).

In the Advanced Port Mapping Settings dialog box, make sure SSL is enabled and select the correct version.

Very few events are being recorded

- **Problem:** The events counter shows less than five events, while the application keeps getting data from the server.
- **Possible Cause:** VuGen's recording mechanism is unable to capture activity over the network.
- **Solution:** Ensure that your application really does provide some network traffic, that it actually sends and receives data through the IP network.

If an antivirus program is running, turn it off during recording. Check the recording log for any clues about the recording failure. Messages such as "connection failure" or "connection not trapped" can be a sign of the incorrect Port Mapping settings.

In addition, if you are recording on a Chrome or Firefox browser, make sure that all the instances of the browser are closed prior to recording.

Specific events are not being recorded

- **Problem:** Certain events are not being recorded.
- **Possible Cause:** VuGen has classified this event as an insignificant action. By default, the Web - HTTP/HTML protocol only records client requests that return an HTTP response status of 2xx or 302, and discards all other requests. If a request returns a response that was discarded, such as 301, VuGen will not generate a step.
- **Solution:** Modify the registry to include the missing status. Locate the following registry key:

```
[HKEY_CURRENT_USER\Software\Mercury  
Interactive\Networking\Multi Settings\QTWeb\Recording]
```

Add the following string value to it:

```
"GenerateApiFuncForCustomHttpStatus"="301"
```

Application hangs during recording

- **Problem:** The recorded application becomes unresponsive during the recording.
- **Possible Cause:** VuGen's recording mechanism is unable to connect to the application's server.
- **Solution:** Check the Recording Log in the Output pane (select Recording from the drop down list of logs) for a message about the Request Connection: Remote Server.

Open the Port Mapping dialog box (**Record > Recording Options > Network > Port Mapping > Options**) and add an entry (**New Entry**) for the application's server. Clear the check box adjacent to that entry. This will ensure that the above IP and port are not recorded—the application connects to them without any LoadRunner involvement.

If communication with the server is essential to the business process, keep the entry checked.

Wrong Server Certificate

- **Problem:** During the recording the recorded application shows an error message about a wrong server certificate.
- **Possible Cause:** The problem is caused by the inability of the client side to verify the validity of the server certificate.
- **Solution:** The LoadRunner Certificate Authority (CA) file should be added to the machine's "Trusted Root Certificate Authorities" certificate store (in case of a Java client application, LoadRunner's CA should be added to Java's trusted CA list using the keytool). This file is provided with LoadRunner, and is called **wplusCAOnly_Expiration_2022.crt**, located in the <LR_folder\bin\certs> folder.

To add it to the store, double-click on the file to open the certificate. Then click **Install Certificate...** to open the Certificate Import Wizard. Use the **Place all certificates in the following store** option and select **Trusted Root Certification Authorities**. When the wizard is completed, you should be able to record the application.

Browser crashes during Ajax Click and Script Recording

- **Problem:** The browser crashes while recording the Ajax Click and Script protocol.
- **Possible Cause:** This may happen when some of the Ajax controls inside the application are not recognized by VuGen.
- **Solution:** Go to the <LR_folder\dat\protocols> folder and open the **WebAjax.lrp** file. Comment out (put a semi-colon (;) in front of the following line:
`DllGetClassObject:jscript.dll=DllGetClassObjectHook:ajax_hooks.dll`



See also:

- For more updates, see the blog post at <http://community.hpe.com/t5/LoadRunner-and-Performance/5-tips-to-solve-the-most-common-problems-seen-in-Web-recording/ba-p/6357387>.

Troubleshooting and Limitations - Web - HTTP/HTML Protocol



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

- Certain POST requests may require HTTP headers which are not automatically generated in the recorded script. To add headers, use the `web_add_header` API function. For details, see the Function Reference (**Help > Function Reference**).
- Port mapping configurations is not supported in the Proxy Recording mode.
- VuGen cannot get a client certificate from Internet Explorer 10 while recording a session.
Workaround: Provide a client certificate in the port mapping settings.

- When strong private key protection is set on a certificate and the WinInet mode is used during the replay, you may be required to manually enter authentication details when replaying the script.
- In previous versions of LoadRunner, the C type "char" was considered a "signed char". In LoadRunner 11.50 and later, it is considered as an "unsigned char". If you used "char" without specifying whether it is signed or unsigned, and performed arithmetic operations on this variable, the results may be different when comparing current results with those from previous versions of LoadRunner.
- When using certificates, if your script refers to the certificate by a wrong index number, such as **web_set_certificate_ex("CertIndex=2", LAST)**, the authentication may fail.
Solution: Regenerate the script (not for WinInet mode recording), or manually set the correct certificate index. To determine this, in Internet Explorer, open the Certificates list (**Tools > Internet options> Content > Certificates**) and use the order of the certificates as they are listed, beginning with 1.
- When recording a Java application with SSL, you need to install the LoadRunner CA certificate. To install the certificate:
 - a. Copy the certificate file **wplusCAOnly_Expiration_2022.crt** from the **<LR_Installation>\bin\certs** folder, to the location of the Java keystore file, **cacerts**. This example assumes the typical folder, **C:\Program Files (x86)\Java\jre6\lib\security**.
 - b. Rename the file to **wplus.crt**
 - c. Open the command prompt, and go to the keystore file's folder, by typing `cd C:\Program Files (x86)\Java\jre6\lib\security`.
 - d. Run `C:\Program Files (x86)\Java\jre6\bin\keytool -list -keystore cacerts` (no quotes).
 - e. You should see the list of all installed certificates. When prompted for a password, enter **changeit** (no quotes).
 - f. Run `C:\Program Files (x86)\Java\jre6\bin\keytool -import -file wplus.crt -keystore cacerts -alias HPCA --trustcacerts` (no quotes). When prompted for a password, enter **changeit** (no quotes).
 - g. Record your secure application.



See also:

- ["Troubleshooting and Limitations for Recording with VuGen" on page 229](#)
- ["Common Web Recording Problems" on page 688](#)

Web Protocols (Generic)

The **Web Protocols** section includes information that is generic to all Web Vuser protocols. For information that is specific to a given Web Vuser protocol, see the documentation for that specific protocol.

Web Protocols - Overview

Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on the next page](#).

You use VuGen to develop Web Vuser scripts. While you navigate through a site performing typical user activities, VuGen records your actions and generates a Vuser script. When you run the script, the resulting Vuser emulates a user accessing the Internet.

Suppose you have a web site that displays product information for your company. The site is accessed by customers and potential customers. You want to make sure that the response time for any customer query is less than a specified value (for example, 20 seconds)—even when many users (for example, 200) access the site simultaneously. You use Vusers to emulate this scenario, where the web server services simultaneous requests for information. Each Vuser could do the following:

- Load the home page
- Navigate to the page containing the product information
- Submit a query
- Wait for a response from the server

You can distribute several hundred Vusers among the available testing machines, each Vuser accessing the server by using its API. This enables you to measure the performance of the server under the load of many Vusers.



See also:

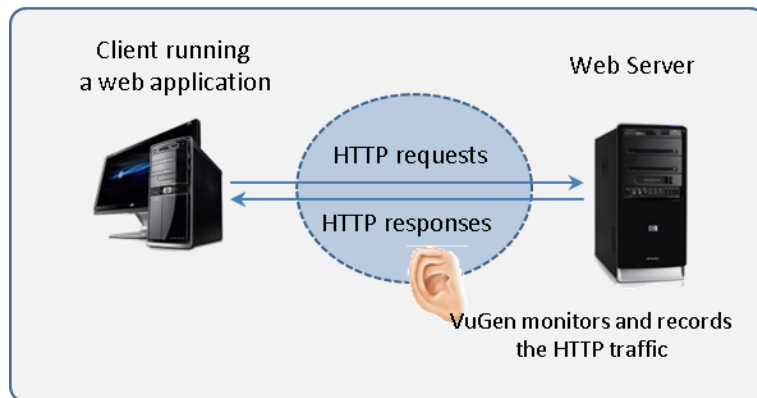
- ["Web Vuser Technology" below](#)

Web Vuser Technology

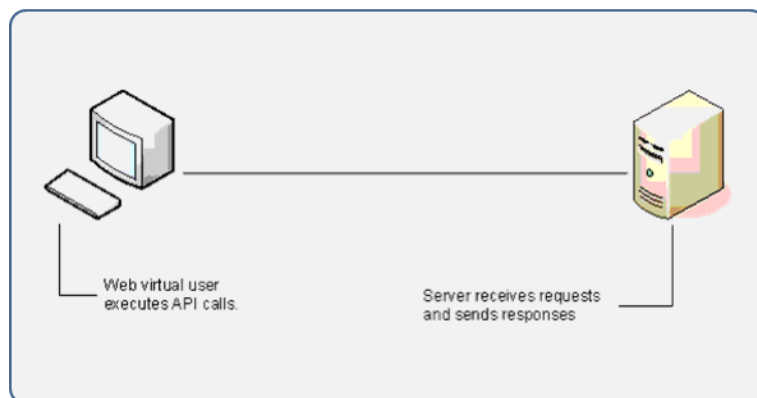
Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on the next page](#).

VuGen creates Web Vuser scripts by monitoring and recording the web traffic that flows between a web browser and a web server while you perform typical business processes. The web traffic includes HTTP requests sent by the browser to the server, and the HTTP responses returned by the server.



When you run a Web Vuser script, the resulting Vuser communicates directly with the web server without relying on a browser or client software. To perform this communication, the Vuser script sends web API functions directly to the web server.



Web Vuser Types

LoadRunner enables you to build and run web-based Vuser scripts using a variety of Web Vuser protocols. Following is a list of LoadRunner's Web Vuser protocols:

- Ajax (Click & Script)
- Flex
- Java over HTTP
- Oracle - Web Applications
- SAP - Web
- Siebel - Web
- TruClient - TruClient - Web, TruClient - Native Mobile, TruClient - Mobile Web
- Web - HTTP/HTML

Text and Image Verification (Web Vuser Scripts) - Overview

Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on the previous page](#).

VuGen enables you to add checks to your Web Vuser scripts. A check verifies the presence of a specific object in a Web page. The object can be either a text string or an image.

Checks enable you to determine whether or not your Web site is functioning correctly while it is being accessed by many Vusers—that is, does the server return the correct Web pages? This is particularly important while your site is under the load of many users, a period when the server is more likely to return incorrect pages.

For example, assume that your Web site displays information on the temperatures in major cities around the world. You use VuGen to create a Vuser script that accesses your Web site.

The Vuser accesses the site and executes a text check on this web page. For example, if the word **Temperature** appears on the page, the check passes. If **Temperature** does not appear because, for example, the correct page was not returned by the server, the check fails. Note that the text check step appears before the URL step. This is because VuGen registers, or prepares in advance, the search information relevant for the next step. When you run the Vuser script, VuGen conducts the check on the web page that follows.



Although the server may display the correct page when you record the script and when a single Vuser executes the script, it is possible that the correct page will not be returned when the server is under the load of many Vusers. The server may then be overloaded and may therefore return meaningless or incorrect HTML code. Alternatively, in some instances when a server is overloaded, the server returns a **500 Server Error** page. In both of these cases, you can insert a check to determine whether or not the correct page is returned by the server.

Note: Checks increase the work of a Vuser, and therefore you may be able to run fewer Vusers per load generator. You should use checks only where experience has shown that the server sometimes returns an incorrect page.

See also:

- ["Understanding Web Text Check Functions" on the next page](#)
- ["Add Text Checks and Image Checks \(Web Vuser Protocols\)" on page 696](#)

Understanding Web Text Check Functions

Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

web_reg_find

When you add a text check to a Web Vuser script, VuGen adds a **web_reg_find** function to the script. This function registers a search for a text string in an HTML page. Registration means that the Vuser does not execute the search immediately—rather the Vuser performs the check only after executing the next Action function, such as **web_url**.

Note: If you are working with a concurrent functions group, the **web_reg_find** function is executed only at the end of the grouping.

In the following example, the **web_reg_find** function searches for the text string "Welcome". If the string is not found, the next action function fails and the script execution stops.

```
web_reg_find("Text=Welcome", "Fail=Found", LAST);  
web_url("Step", "URL=...", LAST);
```

In addition to the **web_reg_find** function, you can use other functions to search for text within an HTML page including:

web_find

This function is used primarily used for backward compatibility. **web_find** differs from the **web_reg_find** function in that **web_find** is limited to HTML-based scripts (see **Recording Options > Recording** tab). It also has less attributes, such as instance, allowing you to determine the number of times the text appeared. When performing a standard text search, **web_reg_find** is the preferred function.

web_global_verification

This function lets you search the data of an entire business process. In contrast to **web_reg_find**, which applies only to the next Action function, this function applies to **all** subsequent Action functions such as **web_url**. By default, the scope of the search is NORESOURCE, searching only the HTML body, excluding headers and resources.


web_global_verification is ideal for detecting application level errors that are not included the HTTP status codes. This function is not limited to an HTML-Based script (see **Recording Options > Recording** tab).

Add Text Checks and Image Checks (Web Vuser Protocols)

Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

Add a Text Check While Recording

1. In the web browser, select the desired text.
2. Click the **Insert Text Check** button  on the VuGen Recording toolbar. VuGen adds a **web_reg_find** function to the script.

Note: When recording a multi-protocol script, you can only insert a text check in the Web part of the script.

Add a Text Check After Recording

1. In the Snapshot pane, display a snapshot that contains the text you want to verify.
2. In the Snapshot pane toolbar, click **HTTP Data** to display the HTTP Data view of the snapshot.
3. In the snapshot, select the text you want to verify. The text must be located in a response section of the snapshot - not in a request section.
4. Right-click and select **Add Text Check Step** from the menu.
5. Modify the options in the Find Text dialog box. For details on the dialog box options, press F1 when in the dialog box to open the Function Reference.
6. Click **OK** to insert the function into the Vuser script.

Add Other Text Checks After Recording

1. In the script editor, locate the position where you want to insert the check.
2. Select **View > Toolbox** to open the Toolbox.
 - a. To insert a **web_reg_find** function, in the Toolbox, under **Services**, select **web_reg_find**.
 - b. To insert a **web_global_verification** function, in the Toolbox, under **Services**, select the required **web_global_verification** function.
3. Drag the selected function to the required location in the Editor.
4. Enter the required details in the dialog box that opens. For details on the dialog box options, press F1 when in the dialog box to open the Function Reference.
5. Click **OK** to insert the function into the Vuser script.

Add an Image Check After Recording

1. In the Editor, locate the position where you want to insert the check.
2. Select **View > Toolbox** to open the Toolbox.

3. In the Toolbox, under **Web Checks**, select **web_image_check**.
4. Drag the selected **web_image_check** function to the required location in the Editor.
5. Enter the required details in the Image Check Properties dialog box. For details on the dialog box options, press F1 when in the dialog box to open the Function Reference.
6. Click **OK** to insert the function into the Vuser script.



See also:

- ["Text and Image Verification \(Web Vuser Scripts\) - Overview" on page 694](#)

Web Snapshots - Overview

Note: Applies to: Web Vuser protocols

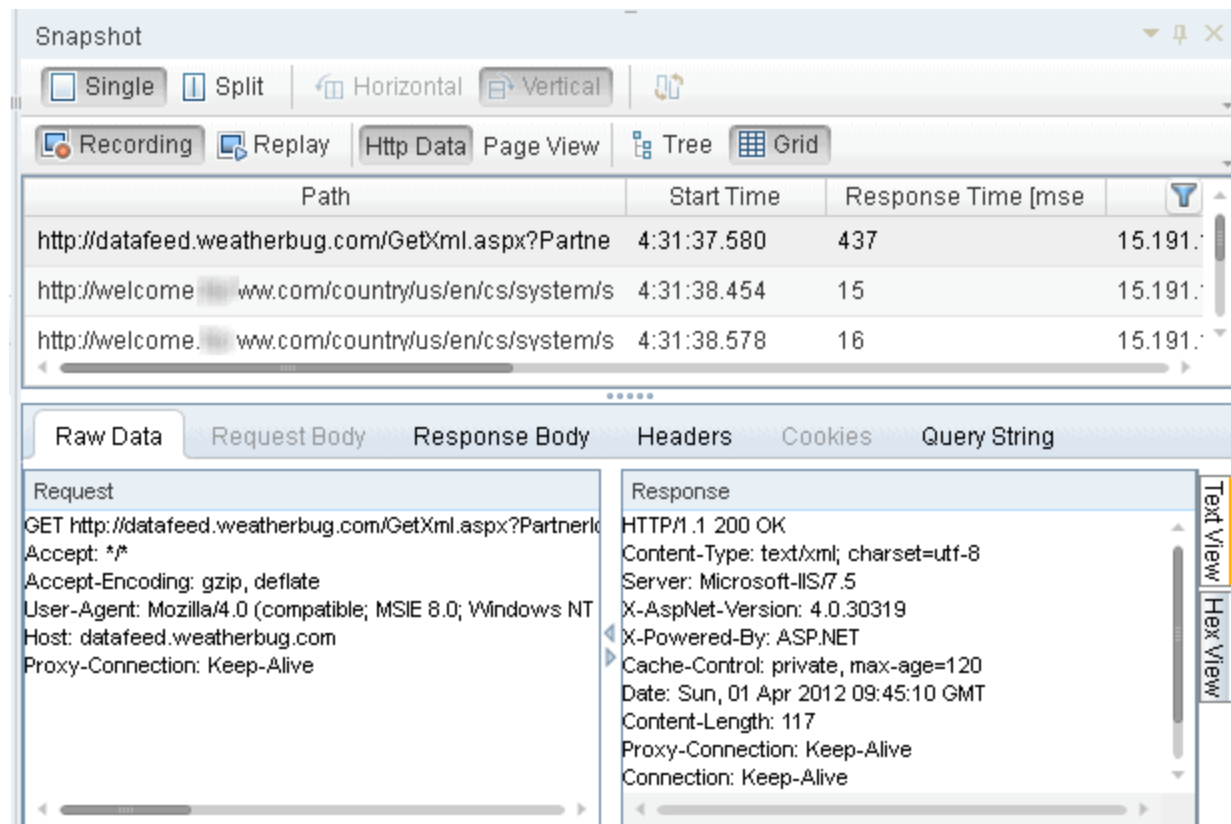
For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

Web Vuser scripts use VuGen's Snapshot pane.

- For details on how to work with the Snapshot pane, see ["Work with Snapshots" on page 279](#).
- For details on the Snapshot pane UI, see ["Snapshot Pane" on page 62](#).


The snapshots show detailed information about some of the steps in the Vuser script. Each snapshot can be displayed using either the **Page** view or the more detailed **HTTP Data** view.

The HTTP Data view displays each HTTP transaction in either a tree view or a grid view. The transaction data is broken up into response data, request data, headers, cookies, and query strings.



Data in the snapshots can be displayed in a number of formats: Data view, Text view, and Hex view.

You can split the Snapshot pane to display two snapshots - typically a record snapshot and the corresponding replay snapshot. If both snapshots are displayed in the HTTP Data view, you can click

the Sync button  on the Snapshot pane toolbar to synchronize the data that is displayed in the two snapshots. For details, see ["Work with Snapshots" on page 279](#).

Correlations and parameters can be created on response data by selecting the desired text and right-clicking.

For data that is difficult to work with (such as binary data), VuGen offers a variety of Data Format Extensions that can transform certain data types into more readable formats. Data that has been formatted by a Data Format Extension can be displayed in its original or formatted state. For more information, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Browser Emulation - Overview

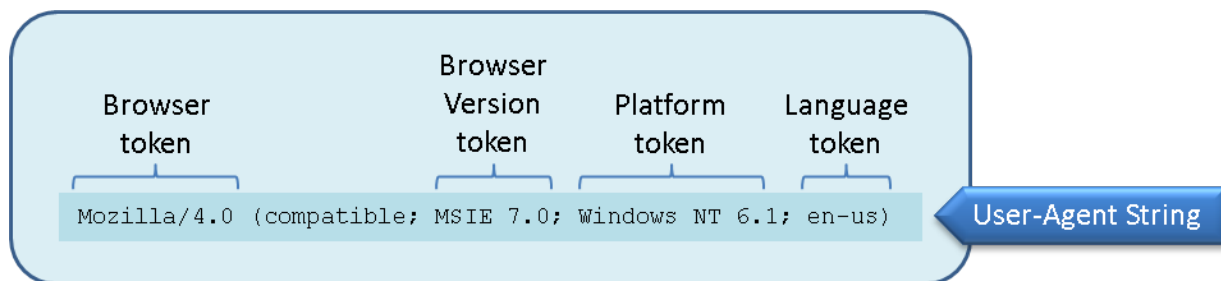
Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

When you run a Web Vuser that accesses a web-site, the Vuser does not use an actual browser to access the site. Instead, the Vuser *emulates* a browser accessing the site. To enable the emulation, the Vuser uses a user-agent string.

What is a User-Agent String?

When a browser sends a request to a server, the browser sends a **user-agent string** that identifies itself to the server. The identifying details in the user-agent string are included in various **tokens**. These tokens provide various details such as which browser is being used, the version of the browser, and the operating system on which the browser is running.



The user-agent string is included in a **User-Agent** header that is part of every request that is sent by the browser to the server. Servers can use the information that is contained in the user-agent string to provide content that is tailored for the specific browser.



Below is an example of a User-Agent header that contains a user-agent string that is sent to a server. In this example, the user-agent string identifies the browser as Internet Explorer 7.0, running under Windows 7.

```
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1)
```

Emulating a specific browser

When you run a Web Vuser that accesses a web-site, the Vuser *emulates* a browser accessing the site. To enable the emulation, the Vuser creates a user-agent string that includes various tokens.

- The *Browser* and *Version* tokens define the browser to emulate and the version of the emulated browser.
- The *Platform* token defines the operating system on which the emulated browser is running.
- The *Language* token defines the language for which the emulated browser is localized.

The user-agent string is included in a **User-Agent** header in every request that the Vuser sends to the server that hosts the web-site.

What information is included in Web Vuser's user-agent string?

1. When a Vuser script is first created, the browser in the user-agent string is Internet Explorer 6, and the operating system is Windows.
2. After the code in a Vuser script is generated, the browser in the user-agent string is changed to the browser that was used when the script was recorded, and the operating system is changed to the operating system that was used when the script was recorded.

After the code in a Vuser script is generated, you can use the Vuser script's runtime settings to specify an emulated browser that is different from the browser that was used when the script was recorded. There are scenarios in which this enables you to more accurately emulate the real-world situation. When you specify the emulated browser, you can specify the browser type and version, the platform on which the Vuser runs, and the language for which the browser is localized. VuGen creates a user-agent string that includes the details that you specify. If required, you can edit the user-agent string to create a customized user-agent string.

To specify a different browser to emulate:

1. Open the Vuser script.
2. Click **Replay > Runtime Settings**.
3. In the runtime settings dialog box, click **Browser > Browser Emulation**, and then select **Use Browser**.
4. Select from the lists of available options to specify the browser to emulate.

Note: The **User-Agent String** that is displayed is updated after each selection that you make.

To customize the user-agent string:

1. Open the Vuser script.
2. Click **Replay > Runtime Settings**.
3. In the runtime settings dialog box, click **Browser > Browser Emulation**, and then select **Use Custom**.
4. Modify the user-agent string as required.

Maximum number of concurrent connections

When a browser accesses a web-site, the browser maintains a number of concurrent connections with the web server that hosts the web-site. Therefore, when you access a web page that contains many different objects, such as images, Javascript files, frames, data feeds etc, the browser may try to improve performance by downloading several of the objects simultaneously. The maximum number of concurrent connections is dependent on the browser type, and the version of the browser. For example,

Internet Explorer 6 limits the number of concurrent connections to 2; Internet Explorer 8 and Firefox 3+ limit the number of concurrent connections to 6.

When you run a Web Vuser, by default, the maximum number of concurrent connections that the Vuser can maintain with a web-server is defined by the browser that is specified in the Vuser script's user-agent string.

To override the default maximum number of connections value:

You can use the **MAX_CONNECTIONS_PER_HOST** option in the **web_set_sockets_option** function. For details, see the Function Reference (**Help > Function Reference**).

The following example sets the maximum number of concurrent connections to 10:

```
web_set_sockets_option("MAX_CONNECTIONS_PER_HOST", "10");
```

Note: When you run a Web Vuser script, the maximum number of concurrent connections that the Vuser can maintain with the server appears in the Replay log.

Perform Load Testing with nCipher HSM

This task describes how to set up a script to load test an environment with an nCipher HSM (Hardware Security Module).

1. **Prerequisite:** Generate the client certificate file (client_cert.pem) with the private key (client_key.pem) pointing to the private keys stored in the HSM. Make sure that you can connect to your Web server (A.com) with a generated CA file (ca-certs.pem). The successful openssl command should have the following form:

```
openssl s_client -connect A.com:443 -CAfile ca-certs.pem -cert client_cert.pem -certform PEM  
-key client_key.pem -keyform PEM -engine CHIL
```

2. Set up the **PATH** environment variable:
 - a. Add the **nfhwcrhk.dll** file, usually located in *C:\Program Files (x86)\nCIPHER\nfast\toolkits\hwcrhk*, to the PATH environment variable.
 - b. Restart VuGen to apply the changes.
3. Enable nCipher key retrieval
 - a. In the runtime settings, go to the **Internet Protocol > Preferences** view and expand the **Authentication** section.
 - b. Select the **Enable retrieving keys from nCipher HSM** option. For details, see [Preferences View - Internet Protocol](#).
4. Edit the Vuser script
Add the following content to the **vuser_init** section of your Vuser script.

```
web_set_sockets_option(SSL_VERSION, "TLS");  
web_set_sockets_option(DEFAULT_VERIFY_PATH, <full_path>\ca-certs.pem);  
web_set_certificate_ex (  
    "CertFilePath=<full_path_to_client_cert_file>/client_cert.pem",  
    "CertFormat=PEM",  
    "KeyFilePath= <full_path_to_client_private_key_file> /client_key.pem",  
    "KeyFormat=PEM",  
    LAST);
```

Working with Cache Data

Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

Web browsers maintain a cache of objects that are downloaded by the browser. This lets the browser load objects faster when accessing a website because it uses objects from the cache instead of downloading the objects again.

When you run a Vuser script, by default, the Vuser starts with an empty cache. This implies that the Vuser must download each object the first time that the object is required. To better emulate a real-world situation, you can provide the Vuser with a pre-defined cache at any stage while the Vuser is running. The Vuser can then access required objects directly from the cache, without having to download them.

To provide a Vuser with a pre-defined cache, first you create a file that contains the cache, and then you load the cache into the Vuser. You use the **web_dump_cache** function to create the cache file, and the **web_load_cache** function to load the cache file into the Vuser.

Create the cache file

You use the **web_dump_cache** function to create a cache file. The **web_dump_cache** function creates a file that contains all the objects that exist in the Vuser cache when the **web_dump_cache** function is executed. Insert the **web_dump_cache** function into a Vuser script, typically towards the end of the *Action* section of the script. This will ensure that the Vuser cache contains the required objects when the **web_dump_cache** function is run to create the cache file. After inserting the **web_dump_cache** function, run the script to build the Vuser cache and create the cache file.

Note: You need to run the **web_dump_cache** function only once to generate the required cache file. Typically, this run is not part of a scenario. After the cache file has been created, when you run the Vuser script as part of a scenario, there is no need to execute the **web_dump_cache** function. Therefore you should comment-out the **web_dump_cache** function in the Vuser script.

You use the **FileName** argument in the **web_dump_cache** function to specify the name and location of the file to create. The **FileName** path can be either absolute (e.g. "FileName=c:\\MyDir\\User1.cache") or relative to the current Vuser directory (e.g. "FileName=Iteration1.cache").

- **Absolute path names:** Use absolute path names if you do not want the cache file to be linked to the script. For example, if you wish to use a different cache on each host, use an absolute path.
- **Relative path names:** If you use a relative path name, the cache file is created inside the Vuser directory. When you copy the Vuser script, save it to a new location, or copy it to a load generator host, the cache file is also copied. Relative paths are independent of drive mappings and network locations.

The file extension of the cache file is always ".cache". The ".cache" extension is added if it is not specified in the **FileName** argument. For example, if you specify "FileName=Iteration2.txt", the cache file is called "FileName=Iteration2.txt.cache".

File names in the **FileName** argument can be parameterized so that different Vusers or different iterations can use different cache files. For example, "FileName=Iteration{param}.cache"

In the following example, the **web_dump_cache** function creates a cache file for each VuserName parameter running the script. The cache files are located in c:\\temp.



Example:

```
web_dump_cache("paycheckcache","FileName=c:\\temp\\{Vuser  
Name}paycheck", "Replace=yes", LAST)
```

If you run a single Vuser user ten times, VuGen creates ten cache files in the following format, where the "Kunnn" prefix is the VuserName value:

```
Ku001paycheck.cache  
Ku002paycheck.cache  
Ku003paycheck.cache  
...
```

Load the cache file into a Vuser

The **web_load_cache** function loads a cache file into a Vuser. The **FileName** argument in the **web_load_cache** function specifies the name and location of the cache file to load. The specified cache file must exist before the **web_load_cache** function is executed. Therefore, you can run the **web_load_cache** function only after running the **web_dump_cache** function to create the cache file.

In the following example, the **web_load_cache** function loads the **{VuserName}paycheck** cache files from **c:\\temp**.



Example:

```
web_load_cache("ActionLoad","FileName=c:\\temp\\{VuserName}paycheck",LAST)
```

See also:

- For details on cache functions, see the [Function Reference \(Help > Function Reference\)](#).
- For details on how to implement cache functions, see ["Insert Caching Functions" below](#).

Insert Caching Functions

Note: Applies to: Web Vuser protocols

For a list of Web Vuser protocols, see ["Web Vuser Types" on page 693](#).

Caching functions allow you to create a cache file that contains the Vuser cache, and then to load the cache file data into a Vuser.

1. Insert the **web_dump_cache** function into your Vuser script.
2. Run the script to create the cache file.
3. Insert the **web_load_cache** function into your script - before the Vuser actions.
4. Comment-out the **web_dump_cache** function.
5. Run and save the script.

See also:

- Working with Cache Data
- Function Reference ([Help > Function Reference](#))

Data Format Extensions (DFEs) - Overview

Data Format Extension support—or DFE support for short—enables easier scripting of web applications by providing the ability to decode and encode formatted data that is exchanged between the client and the server. This enables easier correlation and parameterization of the generated Vuser scripts.

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

When to use DFEs

When you record a Vuser script, VuGen records the HTTP requests and responses that are passed between the client and the web server. The data in these HTTP requests and responses is often encoded. For example, some of the data may be in binary format. The encoded data may be in the HTTP query string, headers, body, or cookies. When the encoded data is included in a Vuser script, the

resulting script will contain data that is difficult to decipher. This makes it difficult to identify text strings that can be used for parameterization and correlation.

The script segment below shows a section of a Vuser script that was generated while recording business processes on a GWT-based application. Notice how some sections of the script contain encoded data and are therefore difficult to decipher.

```
web_custom_request("gwtService",
  "URL=http://lazarus.devlab.ad:8081/GwtComplexObject/org.ega.Main/sampleService/gwtService",
  "Method=POST",
  "Resource=0",
  "RecContentType=application/json",
  "Referer=http://lazarus.devlab.ad:8081/GwtComplexObject/",
  "Snapshot=t3.inf",
  "Mode=HTML",
  "EncType=text/x-gwt-rpc; charset=utf-8",
  "Body=6|0|7|http://lazarus.devlab.ad:8081/GwtComplexObject/org.ega.Main/|99EB9620EEB0D48791FBC5BF958C6366|",
  "org.ega.client.sampleService.GWTService"
  "|myMethod|org.ega.client.data.InputData/74817998|LoadRunner|11.52|1|2|3|4|1|5|6|7|",
  LAST);
```

LoadRunner uses data format extensions (DFEs) to resolve the difficulties that arise from encoded data in Vuser scripts. DFE support allows easier creation of Vuser scripts by providing the ability to decode the encoded data that is exchanged between the client and the server. By providing the decoded format of the data, the information is presented in the Vuser script in a readable format that enables you to correlate and parameterize the script as required. When the script is replayed, the DFE support re-encodes the modified Vuser script, and enables the Vuser to send the correctly encoded request to the server.

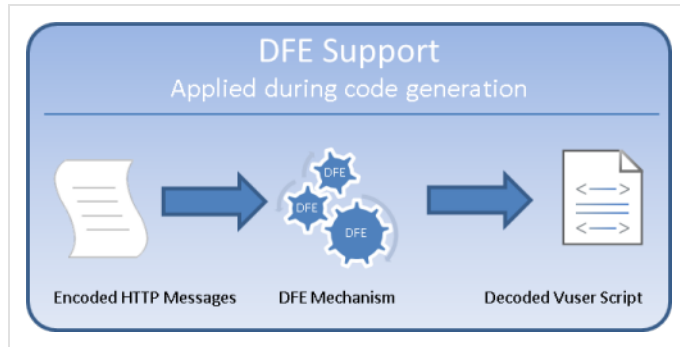
LoadRunner includes a number of pre-defined DFEs. Each DFE is able to decode and encode a specific type of data. For example, the GWT DFE decodes GWT data to XML format when a script is generated, and it encodes XML-formatted data to GWT-formatted data before the script is replayed. For a full list of the pre-defined DFEs, see ["Data Format Extension List" on page 713](#).

When a DFE is applied to a Vuser script and the script is then regenerated, the DFE modifies the script and replaces the encoded data with decoded data. For details on how the script is modified, see ["How DFEs Modify a Vuser Script" on page 712](#).

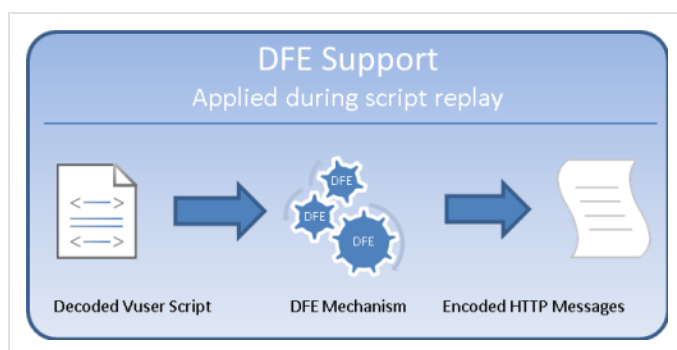
DFE support

You enable DFE support for each Vuser script that requires decoding of encoded data. When DFE support is enabled, the DFE support is applied in the following circumstances:

- Each time the script is generated (after recording) or regenerated. The DFEs are applied to decode the encoded data to produce a script that is easy to decipher.



- Each time the script is run. The DFEs are applied to re-encode the decoded data to produce HTTP messages with encoded data in a format that is expected by the server.



Note: In addition to applying DFEs when a script is generated or replayed, it is possible to apply a DFE to a selected string in a Vuser script. For details, see ["Applying DFEs to a String" on page 714](#).

In some scenarios, decoding of encoded data must be performed in a number of stages, until the fully decoded data is produced. Each stage in the conversion process is performed by applying a specified DFE. For example, encoded data from a response may be decoded by applying three DFEs - first DFE-1, then DFE-2, and then DFE-3. In each stage, the output from one DFE is the input to the next DFE, until the fully decoded data is produced.



DFE chains

The series of DFEs that are required to decode encoded data is defined in a **chain**. For example, you could create a chain called **DFE-Chain-1** that includes three DFEs: DFE-1, DFE-2, and DFE-3. The

sequence of the DFEs inside a chain is significant—the sequence indicates the order in which the DFEs are applied to the encoded data.



If only a single DFE is required to decode encoded data, the DFE must still be included in a chain.



Assigning DFE chains

HTTP messages can be divided into a number of sections, including a body, headers, cookies, and a query string. After you define the DFE chains that will be applied to decode and encode a Vuser script, you must specify to which sections of the HTTP messages the DFE chains will apply. Because each HTTP message has only one **Body** section and one **Query String** section, you can specify only a single DFE chain to apply to each of these sections. In contrast, each HTTP message can contain numerous headers and cookies. Consequently, you can specify a particular DFE chain to apply to each header and cookie. For details, see ["Apply DFE Chains to Sections of the HTTP Message" on page 711](#).

Replaying Vuser scripts that contain DFEs

When you replay a Vuser script that contains DFE functionality, various messages are added to the Replay log in VuGen's Output pane. Make sure to check these messages to ensure that the DFE functionality is correctly implemented. For further details, see ["Troubleshooting - Data Format Extension \(DFE\)" on page 724](#).

See also:

- ["Implement Data Format Extension \(DFE\) Support" below](#)
- For details on creating custom DFEs (advanced users only), see [HPE Software Support Knowledge Base](#).
- For details on using the VuGen JavaScript Engine to encode and decode data using common JavaScript libraries, see ["Using the VuGen JavaScript Engine" on page 669](#).

Implement Data Format Extension (DFE) Support

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

To implement the DFE support for a Vuser script, you must perform the steps shown in the diagram below:



You may encounter the following errors with DFE functions on 64-bit Linux environments:

Error -27040: Data Format Extension: Creating custom chain failed: Extension "UrlEncoding" was not found.

Error -35063: The "DFEs" argument is invalid. Check that the provided extensions have their configuration files defined.

Solution: Install the 32-bit version of keyutils-libs.so (keyutils-libs.i686) on your system, if it does not already exist.

For additional details about DFEs, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Define a Chain of DFEs


Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts


In order to implement DFE support for a Vuser script, you must first define the DFE chains that are applied by the Vuser. Defining the DFE chains is the first step in implementing DFE support.



Adding a DFE chain

1. In VuGen, open the Vuser script.
2. Click **Record > Recording Options > Data Format Extension > Chain Configuration**.
For details on the dialog box options, see ["Data Format Extension > Chain Configuration Recording Options" on page 157](#).
3. Under **Chains**, click the **New Chain** button  and create a new chain. The new chain is listed in the **Chains** pane.

Adding DFEs to the new DFE chain

1. In the **Chains** pane, select the chain to which you want to add DFEs.
2. In the **Chain: <chain name>** pane, click the **Add DFE** button . For details on the **Chain: <chain name>** pane, see ["Data Format Extension > Chain Configuration Recording Options" on page 157](#).
3. In the Add Data Format Extension dialog box, select the DFE that you want to add to the chain and click **OK**.

When you add the **GWT** DFE or the **Prefix Postfix** DFE to a chain, you must supply additional configuration details about the DFE. For more information, see the documentation about the specific DFE.

4. After you add a DFE to the chain, select the appropriate option from the **Continue Processing** list. For details on the **Continue Processing** option, see ["Data Format Extension > Chain Configuration Recording Options" on page 157](#).
5. Add additional DFEs to the chain as required.

See also:

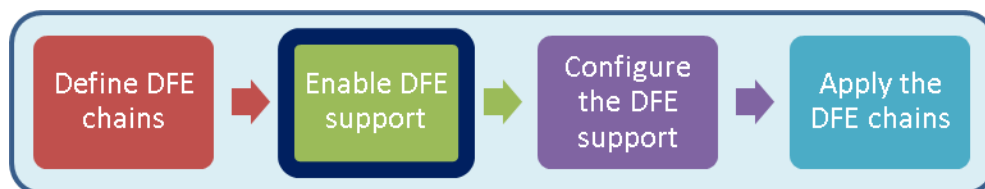
- ["Data Format Extensions \(DFEs\) - Overview" on page 704](#)
- ["Enable DFE Support" below](#)
- ["Implement Data Format Extension \(DFE\) Support" on page 707](#)

Enable DFE Support

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

After you define the DFE chains that are available to a Vuser script, you must enable the DFE support, as described below. Enabling DFE support is the second step in implementing DFE support.



1. In VuGen, open the Vuser script.
2. Click **Record > Recording Options > Data Format Extension > Code Generation**.
For details on the dialog box options, see ["Data Format Extension > Chain Configuration Recording Options" on page 157](#).
3. Select the **Enable data format extension** check box.

See also:

- ["Configure DFE Support" below](#)
- ["Define a Chain of DFEs" on page 708](#)
- ["Implement Data Format Extension \(DFE\) Support" on page 707](#)
- ["Data Format Extensions \(DFEs\) - Overview" on page 704](#)

Configure DFE Support

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

After you enable DFE support for a Vuser script, you can configure the DFE support as described below. Configuring DFE support is the third step in implementing DFE support.



1. Open the Vuser script in VuGen.
 2. Click **Record > Recording Options > Data Format Extension > Code Generation**.
For details on the dialog box options, see ["Data Format Extension > Code Generation Recording Options" on page 161](#).
 3. Under **Configuration**, from the **Format** list, select the parts of the Vuser script to which the DFEs will be applied.
 - **Code and snapshots**. Applies DFEs to convert the Vuser script code and the snapshot data.
 - **Snapshots**. Applies DFEs to convert the snapshot data only - not the Vuser script code.
 4. Select the **Verify formatted data** check box to check the results of the data conversion by converting the converted data back to the original state, and then verifying that it matches the original data.
- For details on the dialog box options, see ["Data Format Extension > Code Generation Recording Options" on page 161](#).
 - After configuring the DFE support, you can assign the DFE chains to specific sections of the HTTP messages. For details, see ["Apply DFE Chains to Sections of the HTTP Message" on the next page](#).
 - For an overview of the process of implementing DFE support, see ["Implement Data Format Extension \(DFE\) Support" on page 707](#).
 - For additional details about DFEs, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Apply DFE Chains to Sections of the HTTP Message

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

After you have defined the DFE chains that are available to a Vuser script, enabled and configured DFE support, you must define to which sections of the HTTP messages to apply the DFE chains, as described below. Applying DFE chains to message sections is the last step in implementing DFE support.



1. Open the Vuser script in VuGen.
2. Click **Record > Recording Options > Data Format Extension > Code Generation**.
For details on the dialog box options, see ["Data Format Extension > Chain Configuration Recording Options" on page 157](#).
3. [Optional] In the **<message section>** pane, click **Body** add then select the chain that will be applied to the message body.
4. [Optional] In the **<message section>** pane, click **Headers** add then select the chains that will be applied to the message headers.

Note: For VuGen to correctly assign the chain to a specific header, the entry in the **Name** column must be exactly the same as the name of the header in the message.

5. [Optional] In the **<message section>** pane, click **Cookies** add then select the chains that will be applied to the message cookies.

Note: For VuGen to correctly assign the chain to a specific cookie, the entry in the **Name** column must be exactly the same as the name of the cookie in the message.

6. [Optional] In the **<message section>** pane, click **Query String** add then select the chain that will be applied to the message query string.

Note: Whereas you can modify only the default chain for the **Body** and **Query String** sections, you can add multiple chains for the **Headers** and **Cookies** sections.

See also:

- ["Implement Data Format Extension \(DFE\) Support" on page 707](#)
- ["Data Format Extensions \(DFEs\) - Overview" on page 704](#)

How DFEs Modify a Vuser Script

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

When a DFE is applied to a Vuser script and the script is then regenerated, the DFE causes various modifications to be made to the script, as follows:

- VuGen replaces the original encoded text string with a parameter.
- VuGen inserts a **web_convert_from_formatted** function before the function that contains the new parameter. The **web_convert_from_formatted** function contains the decoded value of the original encoded text.

The script section below shows a **web_custom_request** function that was generated without DFE support. The **Body** tag in the function includes a text string that is base64 encoded, **Body=TW9uZGF5**. Because the value of the **Body** tag is encoded, it is difficult to change its value if required for correlation or parameterization purposes.

```
web_custom_request("echo_post.asp",  
    "URL=http://example.devlab.ad/cgi-bin/temp/echo\_post.asp",  
    "Method=POST",  
    "Resource=0",  
    "RecContentType=text/html",  
    "Referer=",  
    "Snapshot=t1.inf",  
    "Mode=HTTP",  
    "Body=TW9uZGF5",  
    LAST);
```

After applying Base64 DFE support, the value of the **Body** tag in the regenerated script is replaced with a parameter called DFE_PARAM, "**Body={DFE_PARAM}**", as shown below.

```

/*TODO: A Correlation scan needs to be performed.*/
web_convert_from_formatted("FormattedData/EscapedBinary=<HP_EXTENSION name=\"Base64\">Monday</HP_EXTENSION>",
    "TargetParam=DFE_PARAM",
    LAST);

web_custom_request("echo_post.asp",
    "URL=http://example.devlab.ad/cgi-bin/temp/echo_post.asp",
    "Method=POST",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTTP",
    "Body={DFE_PARAM}",
    LAST);

return 0;

```

In addition, the modified code also includes a **web_convert_from_formatted** function. The function indicates that the decoded value of the originally encoded string is **Monday**. It is now simple to change the value from **Monday** to any other day by simply changing the decoded value in the **web_convert_from_formatted** function.

Data Format Extension List

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

The following table lists the pre-defined LoadRunner DFEs (Data Format Extensions). For more information about DFEs, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Data Format Extension	Description
Base64 Extension	Decodes strings that are encoded with a Base64 encoder.
Binary to XML Extension	Transforms Microsoft WCF binary XML into XML format.
GWT Extension	Transforms GWT data to XML format.
Java To XML Extension	Transforms serialized Java objects to XML.

JSON to XML Extension	Transforms JSON data to XML format.
Prefix Postfix Extension	Enables you to cut data from the beginning and/or end of a string which you do not want decoded. You can add and customize as many prefix/postfix extensions as required. Each postfix/prefix extension created should have a unique display name and tag name.
Remedy to XML Extension	Transforms Remedy request data into XML format. Note: This extension does not transform Remedy response data - which is JavaScript code.
URL Encoding Extension	Decodes strings that are encoded with URL encoding format.
XML Extension	Receives data and checks to see if it conforms with XML syntax. This check allows VuGen to perform correlations based on XPath and to display snapshot data in an XML viewer.
XSS Extension	Enables you to test sites that use Cross Site Scripting (XSS) defense code.

Applying DFEs to a String

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

You can apply DFEs to a Vuser script to decode encoded data in the script. You can apply the DFEs:

- to specified sections of the Vuser script, when the script is generated or regenerated. For details, see ["Implement Data Format Extension \(DFE\) Support" on page 707](#).
- to a single encoded string in the Vuser script, as described in this topic.

For an overview of DFEs, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).

Note: After you apply a DFE to a string in a Vuser script, VuGen adds an entry to the **Tasks** tab. The added entry indicates that a correlation scan should be performed. For details on performing a correlation scan, see ["Correlating" on page 232](#).

To apply a DFE to an encoded string in a Vuser script:

1. Open the script in VuGen, and select the encoded text string.
2. Right-click inside the selection, click **Decode with DFE**, and click the name of the chain that contains the DFEs to decode the encoded string. For details on how to define a DFE chain, see ["Define a Chain of DFEs" on page 708](#).

VuGen replaces the selected text with a parameter, and adds a **web_convert_from_formatted** function that contains the decoded equivalent of the originally selected text.

Note: To change the name that VuGen assigns to new parameters, right-click some encoded text in the Vuser script, click **Decode with DFE > Advanced**, and then specify the parameter name in the **Target Parameter** box. VuGen will add a counter to the parameter name, and increment the counter as required.

Google Web Toolkit - Data Format Extension (GWT-DFE) - Overview

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

The Google Web Toolkit (GWT) is a set of tools that allow web developers to create complex JavaScript front-end applications. The code is developed in Java and is open source. GWT supports, among other features: asynchronous remote procedure calls, history management, bookmarking, UI abstraction, internationalization, and cross-browser portability.

The GWT-DFE support helps to generate Vuser scripts for GWT-based web sites that use the GWT-RPC mechanism. When you record a GWT-RPC based web site without enabling the GWT-DFE support, the resulting Vuser script may contain significant amounts of data that is cryptic and therefore difficult to decipher, as shown in the code segment below:

```
web_custom_request("gwtService",
  "URL=http://lazarus.devlab.ad:8081/GwtComplexObject/org.ega.Main/sampleService/gwtService",
  "Method=POST",
  "Resource=0",
  "RecContentType=application/json",
  "Referer=http://lazarus.devlab.ad:8081/GwtComplexObject/",
  "Snapshot=t3.inf",
  "Node=HTML",
  "EncType=text/plain; charset=utf-8",
  "Body=6|0|7|http://lazarus.devlab.ad:8081/GwtComplexObject/org.ega.Main/|99E89620EEB0D48791FBC58F958C6360|"
  "org.ega.client.sampleService.GWTService"
  "|myMethod|org.ega.client.data.InputData/74817998|LoadRunner|11.52|1|2|3|4|1|5|6|7|",
  LAST);
```

The cryptic formatting of the data makes it difficult to identify text strings to be used for correlation, parameterization, and verification.



Tip: The presence of numerous pipes in the recorded data indicates that the recorded site may



be a GWT-based web site that uses the GWT-RPC mechanism.

When you enable GWT-DFE support, VuGen is able to decode much of the complex data in the HTTP responses and requests. This enables VuGen to generate Vuser scripts that contain data in XML format. In addition, the original coded data contains only values, without the associated names of the data fields. After applying GWT-DFE, the resulting XML-formatted data includes both the names and the values of the data fields. The XML-formatted data in the scripts is therefore easier to decipher, making the scripts easier to correlate, parameterize, and use for verification purposes.

To enable VuGen to decode the complex data in the HTTP communication, you must identify the *.war* file that is used by the web application. Occasionally, an application uses *.jar* files. All considerations below about *.war* files apply equally to *.jar* files when used instead of a *.war* file.

The *.war* file contains the logic used by GWT to encode and decode the information in the HTTP communication. VuGen needs access to the *.war* file so that VuGen can perform similar encoding and decoding procedures. Typically, these *.war* files are located on the application server, under the web applications folder.

Note: Make sure that the *.war* file that you associate with the Vuser script is the most up-to-date *.war* file for your application. The *.war* file is changed each time changes are made to the web application. GWT-DFE support will function correctly only if the most up-to-date *.war* file is available.

- For an introduction to using DFEs in Vuser scripts, see ["Data Format Extensions \(DFEs\) - Overview" on page 704](#).
- For details on how to implement DFE support, see ["Implement Data Format Extension \(DFE\) Support" on page 707](#).
- For a full list of the supported DFEs, see ["Data Format Extension List" on page 713](#).

Note: GWT- DFE provides an automatic solution for GWT specific (STRONG_NAME_HEADER) correlations.

What do you want to read about now?

Example of code generated with and without GWT-DFE support:

Original Script - without GWT-DFE Support

```
6|0|11|http://localhost:8081/MyTestApp/testapp/|624C899BB846618A2E7F49092  
8212946|com.test.client.GreetingService|greetServeCompAns|com.test.client.ComplexObject/198661839  
|GWT User|inside object|java.util.HashSet/1594477813|java.lang.String/2004016611|add  
string1|  
string2|1|2|3|4|1|5|5|1001|1999|6|5|321|1234|7|0|8|0|8|2|9|10|9|11|
```

Script after applying GWT-DFE Support

```
<HP_EXTENSION name="GWT_DFE_1">
<com.hp.dfe.GWT__Request>
<moduleBaseURL>http://localhost:8081/MyTestApp/testapp/</moduleBaseURL>
<rpcRequest>
<flags>0</flags>
<method>
<class>com.test.client.GreetingService</class>
<name>greetServeCompAns</name>
<parameter-types>
<class>com.test.client.ComplexObject</class>
</parameter-types>
</method>
<parameters>
<com.test.client.ComplexObject>
<anIntField>1001</anIntField>
<anotherIntField>1999</anotherIntField>
<name>GWT User</name>
<objectInComposingField>
<anIntField>321</anIntField>
<anotherIntField>1234</anotherIntField>
<name>inside object</name>
<stringsSet/>
</objectInComposingField>
<stringsSet>
<string>add string2</string>
<string>add string1</string>
</stringsSet>
</com.test.client.ComplexObject>
</parameters>
<serializationPolicy
```

Script after applying GWT-DFE Support

```
class="com.google.gwt.user.server.rpc.impl.StandardSerializationPolicy"/>
</rpcRequest>
</com.hp.dfe.GWT__Request>
</HP_EXTENSION>
```

Auto-detection of GWT Remote Procedure Calls (RPCs):

When VuGen generates or regenerates a Vuser script, VuGen scans the HTTP headers in the requests that are sent to the server. If VuGen detects both a **x-gwt-module-base** text string and a **x-gwt-permutation** text string in any of these HTTP headers, VuGen displays a warning in the VuGen Error tab. The warning recommends that you enable GWT DFE for the Vuser script.

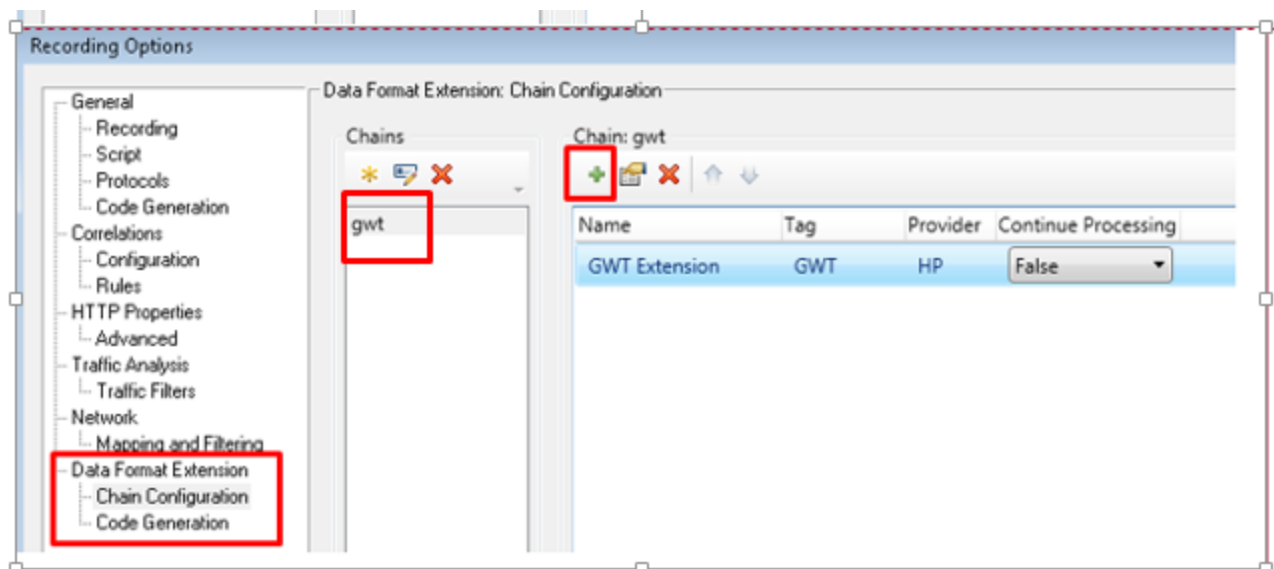


Note: VuGen will continue to issue the above warning - each time the script is generated or regenerated - until the GWT DFE is enabled.

How to apply a GWT Data Format Extension to your script before recording.

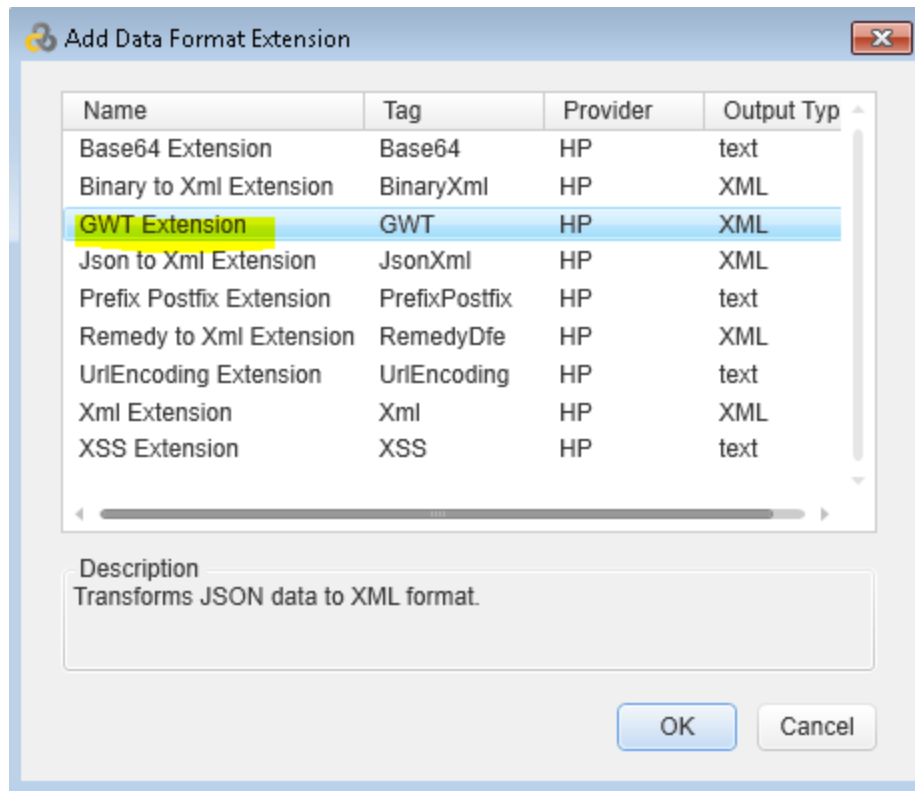
To apply GWT-DEF before recording, perform the following actions:

1. Open Recording Options (VuGen --> File --> Recording options)
2. Go to the "Data Format Extension --> Chain Configuration" tab
3. Add a new chain (press the green '+' sign)



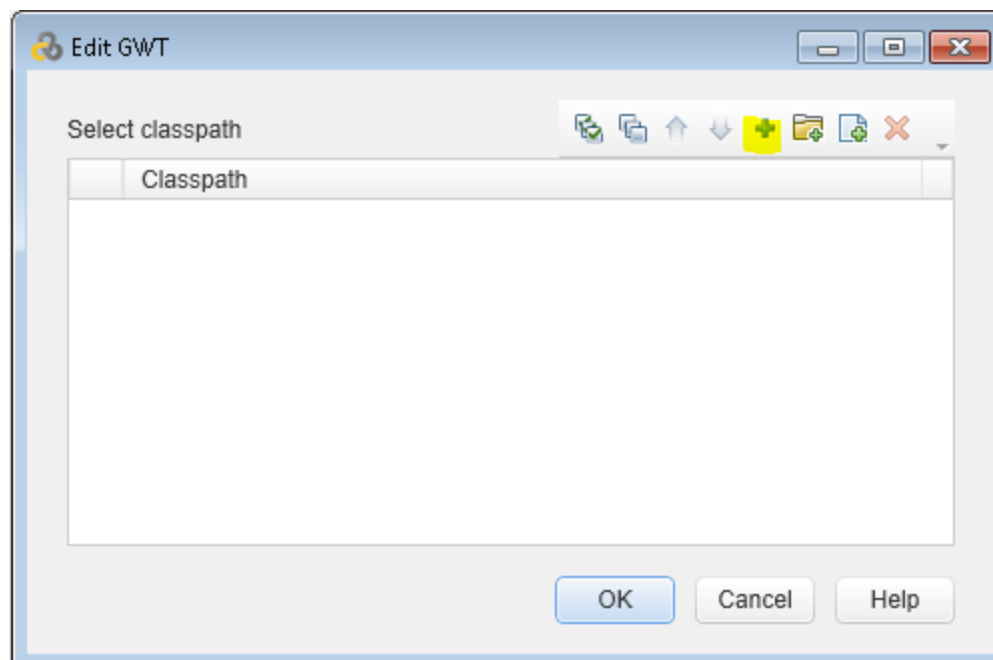
The following dialog will open,

4. Select “GWT Extension” and press OK

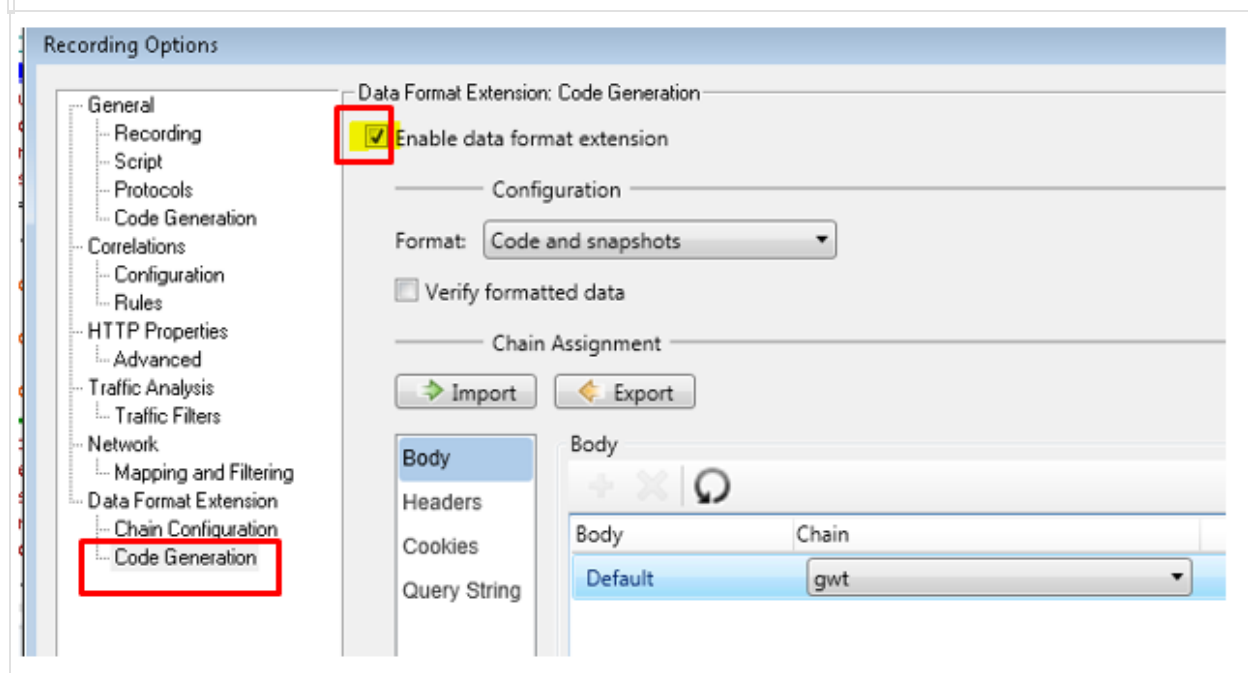


5. A new window will then open.

To add the most recent WAR / JAR files, press the ‘+’ button and add them from the file system.



6. Enable Data Format Extension



7. Record or regenerate the GWT script

After generating the script using GWT DFE, you will get a readable LoadRunner API that includes the request inside an XML.

Tips and troubleshooting

The exact same versions of the war and jar files that are on the server must be used by LoadRunner during code generation and replay.

If there are differences, LoadRunner will give an indication during Code Generation and Replay:

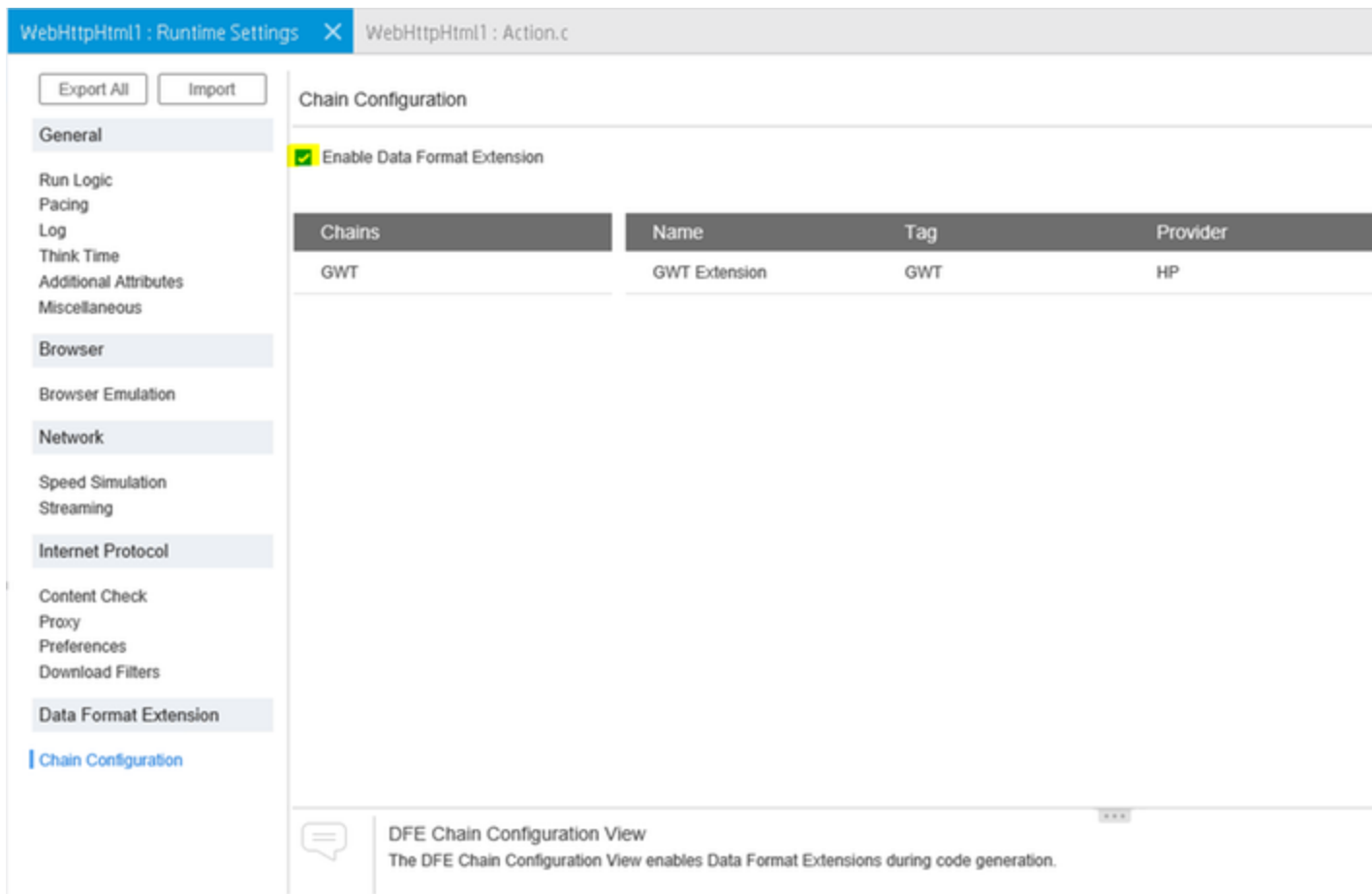
The follow errors can occur if the war and jar files do not match:

- Class not found
- Failed to handle WAR entry.
- Failed to de-serialize method
- Missing method
- The correct GWT serialization policy could not be found.

To fix issues, ensure that:

- The GWT DFE is enabled and the chains are configured correctly, including all .JAR, .CLASS and .GWT.RPC files used by the GWT application. After configuring the Chain, regenerate the script.
- The Data Format Extension is enabled in runtime settings **Data Format -> Chain Configuration ->**

Enable Data Format Extension.



Implementing GWT-DFE Support

Applies to:

- Web - HTTP/HTML Vuser scripts
- Web - HTTP/HTML steps inside Flex Vuser scripts

This topic provides additional information about implementing DFE support, as described in ["Implement Data Format Extension \(DFE\) Support" on page 707](#).

Prerequisites for implementing GWT-DFE support

LoadRunner includes OpenJDK 7 JRE. However, some applications may be compiled for JVM1.8 and higher. If your application is compiled with JDK1.8 or higher, replace the <LoadRunner>\lib\openjdk32\jre folder with your own OpenJDK 8 or higher JRE before recording a GWT application.

Recording GWT-DFE Headers

As part of the GWT-DFE support implementation process, it is necessary to specify that VuGen record **x-gwt-permutation** headers when recording business processes. This procedure, as described below, can be performed at any stage of the GWT-DFE support implementation process.




1. Create a Web - HTTP/HTML protocol script .
2. Select **Record > Recording Options > HTTP Properties > Advanced** and then click **Headers**.
3. In the Headers dialog box, select **Record headers in list**.
4. From the **Headers** list, select the **x-gwt-permutation header** check box.

Applying GWT-DFE chains

When you implement any DFE support, you must apply the DFE chains to specific sections of the HTTP communication. The basic process is described in ["Apply DFE Chains to Sections of the HTTP Message" on page 711](#). This topic includes information required when assigning chains while implementing GWT-DFE support. When you assign the chains while implementing GWT-DFE support, you must specify the classpath entries that are associated with the application that is operated by the Vusers. To assign the classpath entries, you must have access to the GWT WAR folder that is used by your development team. The WAR folder includes the following file types:

- *.gwt.rpc files
- *.jar files
- *.class files

Specifying the classpath entries

1. Select **Record > Recording Options > Data Format Extension > Chain Configuration**.
2. Under **Chains**, click  to create and name a new DFE chain.
3. Click .
4. In the Add Data Format Extension dialog box, select **GWT Extension** and click **OK**.
5. In the **Add GWT** dialog box, specify the classpath entries:
 - a. If the classpath entries are contained in a single .war file, click  and then specify the location of the .war file.




Note: If you have write permissions in the folder containing the .war file, it automatically creates a new folder with the extracted contents. It adds the specific classes/jars to VuGen in the following structure:

```
<ServerDir>\<applicationDir>\<MyGWTApplication>\<SomeDirContaining .gwt.rpc file>
```

- WEB-INF\classes
- WEB-INF\lib\gwt-servlet.jar
- WEB-INF\lib\gwt-servlet-deps.jar
- WEB-INF\lib\log4j.jar
- WEB-INF\lib\<AdditionalAUTRelatedJarFile>.jar

If you do not have write access, it will just add the .war file without extracting its contents.

b. If the classpath entries are not contained in a single .war file:

- Click  to add the folder that contains .gwt.rpc files.
- Click  to add the application **classes** folder.
- Click  to add the application **JAR** files from the **WEB-INF\lib** folder.

Note: If the location of the classpath entries is not the same on the computer on which the script was recorded and the computer on which the script will be replayed, then you must modify the runtime settings for the script. Select **Replay > Runtime Settings > Data Format Extension > Chain Configuration** and specify the location of the classpath entries on the computer on which the script will be replayed.

6. Select **Recording Options > Data Format Extension > Code Generation**.
7. Select the **Enable data format extension** check box.
8. Under **Configuration**, select **Code and snapshots** from the **Format** list.
9. Under **Chain Assignment**, select **Body** and select a chain. Select **Headers** and choose the same chain.
10. Click **OK**.

Troubleshooting - Data Format Extension (DFE)

This section describes troubleshooting tasks for Vuser scripts that contain DFE functionality.

- Data Format Exceptions (DFEs) are not supported on Linux. Make sure that DFE support is not enabled for the script, and that it does not contain any DFE steps, before running on Linux.
- **Replay log: Warning -27040: Data Format Extension: Convert: Empty string returned from extension {Extension name}**

When you replay a Vuser script that contains DFE functionality, various messages are added to the Replay log in VuGen's Output pane. The above message indicates that when the Vuser script was replayed, the result of the specified **web_convert_from_formatted** step in the script was an empty string. For some DFEs, returning an empty string from a **web_convert_from_formatted** step is the

correct behavior. However, if the Vuser script includes GWT-DFE functionality, the above message may indicate one or both of the following:

- Some of the required classpath files are not included in the runtime settings for the Vuser script.
- Some of the required classpath files do not exist in the specified location on the Load Generator.

For details on how to resolve these issues, see ["Implementing GWT-DFE Support" on page 722](#).

- If you have implemented your own version of DFE, the definition of class HTTPEntity in DfeDefinitions.h file has been updated in LoadRunner 11.50. No code change is required, but all DFE extensions need be recompiled.

Web Services Protocol

Web Services - Adding Script Content

Web Service Testing Overview

SOA systems are based on Web Services, self-contained applications that can run across the Internet on a variety of platforms. The services are built using Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). They serve as building blocks, thereby enabling the rapid development and deployment of new applications.

Using VuGen, you create Vuser scripts for testing your SOA environment. You can use a test generation wizard to automatically generate scripts, or you can create the scripts manually.

Adding Web Service Script Content - Overview

Web Services scripts let you test your environment by emulating Web Service clients.

After creating an empty Web Services script, as described in ["Create a New Script Dialog Box" on page 128](#), you add content through one of the following methods: recording, manually inserting Web Service calls, importing SOAP, or by analyzing server traffic.

Recording a Web Services Script

By recording a Web Services session, you capture the events of a typical business process. If you have already built a client that interacts with the Web Service, you can record all of the actions that the client performs. The resulting script emulates the operations of your Web Service client. After recording, you can add more Web Service calls and make other enhancements to the script.

When you record an application, you can record it with or without a Web Service WSDL file. If you include a WSDL file, VuGen allows you to create a script by selecting the desired methods and entering values for their arguments. VuGen creates a descriptive script that can be updated when there are changes in the WSDL.

If you record a script without previously importing a service (not recommended) VuGen creates SOAP requests instead of Web Service call steps. SOAP request arguments are less intuitive and harder to maintain.

For more information, see ["Add Content" on page 733](#).

Adding New Web Service Calls

You can create a Web Services script by manually adding Web Service calls. You design the call based on operation, transport, arguments, and other properties.

For more information, see ["Add Content" on page 733](#).

Importing SOAP Requests

VuGen lets you create Web Service calls from SOAP files. If you have a SOAP request file, you can load it directly into your script. VuGen imports the entire SOAP request (excluding the security headers) with the argument values as they were defined in the XML elements. By importing the SOAP, you do not need to set argument values manually as in standard Web Service calls.

For example, suppose you have a SOAP request with the following elements:

```
- <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <q1:AddAddr xmlns:q1="http://tempuri.org/AddrBook/message/">
    <Addr href="#id1" />
  </q1:AddAddr>
  - <q2:Addr id="id1" xsi:type="q2:Addr"
xmlns:q2="http://tempuri.org/AddrBook/type/">
    <name xsi:type="xsd:string">Tom Smith</name>
    <street xsi:type="xsd:string">15 Elm Street</street>
    <city xsi:type="xsd:string">Pheonix</city>
    <state xsi:type="xsd:string">AZ</state>
    <zip-code xsi:type="xsd:string">97432</zip-code>
    <phone-numbers href="#id2" />
    <birthday xsi:type="xsd:date">1983-04-22</birthday>
  </q2:Addr>
  ...
```

When you import the SOAP request, VuGen imports all of the values to the Web Service call. To view the values, in the Step Navigator, right-click the step and then click **Show Arguments**.

To create a new Web Service call based on a SOAP request, you must first import a WSDL file. If a WSDL is not available, or if you want to send the SOAP traffic directly, you can create a SOAP Request step. You specify the URL of the server, the SOAP action, and the response parameter.

In the Editor, the SOAP Request step appears as a **soap_request** function, described in the Function Reference (**Help > Function Reference**).

For more information, see ["Add Content" on page 733](#).

Analyzing Server Traffic

The main focus when testing enterprises and complex systems, is to measure the performance from the client end. Ordinarily, VuGen records the actions you perform in the application or browser, and generates a script emulating the client actions and requests to the server.

In certain test environments, you may be unable to record the client application to retrieve the requests to the server. This may be a result of the server acting as a client, or because you do not have access to the client application. In these cases, you can create a script using VuGen's **Analyze Traffic** feature.

The **Analyze Traffic** feature examines a capture file containing the server network traffic, and creates a script that emulates requests sent to or from the server.

For more information, see ["Create a Script by Analyzing Traffic \(Web Services\)" on page 737](#).

Script Integration

You can use the completed script to test your system in several ways:

- **Functional Testing.** Run the script to see if your Web services are functional. You can also check to see if the Web service generated the expected values. For more information, see ["Web Services - Preparing Scripts for Replay" on page 753](#).
- **Load Testing.** Integrate the script into a LoadRunner Controller scenario to test the performance of your system under load. For more information, see the *Controller* documentation.
- **Production Testing.** Check your Web service's performance over time through a Business Process Monitor configuration. For more information, see the *HPE Business Process Monitor* documentation.

Web Service Call Attachments

When transferring binary files such as images over SOAP, the data must be serialized into XML. Serialization and deserialization can cause a significant amount of overhead. Therefore, it is common to send large binary files using an attachments mechanism. This keeps the binary data intact, reducing the parsing overhead.

Using attachments, the original data is sent outside the SOAP envelope, eliminating the need to serialize the data into XML and making the transfer of the data more efficient.

The formats used for passing a SOAP message together with binary data are MIME (Multipurpose Internet Mail Extensions) and the newer, more efficient DIME (Direct Internet Message Encapsulation) specifications. VuGen supports DIME for all toolkits, but MIME only for the Axis toolkit. To use MIME attachments for the .NET toolkit, see ["User Handler Examples" on page 760](#).

VuGen supports the sending and receiving of attachments with SOAP messages. You can send Input (Request) or save Output (Response) attachments. For task details, see ["Add Content" on page 733](#).

Output attachments are used to save the response as an attachment. You can choose one of the following options: **Save All Attachments** or **Save Attachment by Index**.

When you specify **Save All Attachments**, VuGen creates three parameters for each attachment based on the parameter name that you specify: a parameter containing the attachment data, the content type of the attachment, and a unique ID for the attachment.

For example, if you specify the name **MyParam** in the **Content** field, the parameter names for the first attachment would be:

MyParam_1
MyParam_1_ContentType
MyParam_1_ContentID

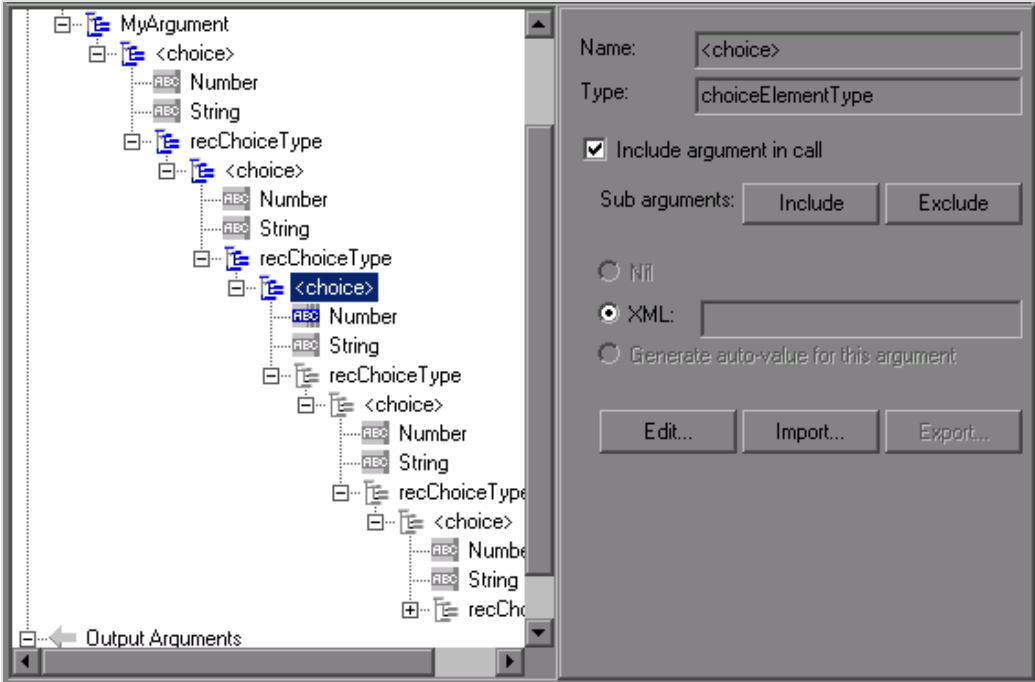
When you specify **Save Attachments by Index**, you specify the index number and name of the parameter in which to store the attachment. The parameter name that you specify for **Content**, is used as a prefix for the Content type and Content ID parameters.

Special Argument Types

VuGen handles special argument types, as follows:

Argument Type	Description
Derived Types	Supported for WSDLs. When setting the properties for a Web Service Call, VuGen allows you to use the base type or derived type for the argument. After you select a type, VuGen updates the argument tree node to reflect the new type. For details, see "New Web Service Call Dialog Box" on page 740 .
Abstract Types	<p>A declaration type declared by the programmer. When an element or type is declared to be abstract, it cannot be used in an instance document. Instead, a member of the element's substitution group, provided by the XML schema, must appear in the instance document. In such a case, all instances of that element must use the xsi:type to indicate a derived type that is not abstract.</p> <p>When VuGen encounters an Abstract type, it cannot create an abstract class and replay will fail. In this case, VuGen displays a warning message beneath the Type box, instructing you to replace the Abstract type with a derived type.</p>

Argument Type	Description
Optional Elements	<p>In WSDL files, optional parameters are defined by one of the following attributes:</p> <pre>minoccurs='0' nillable='true'</pre> <p>minoccurs = 0 indicates a truly optional element that can be omitted. Nillable means that the element can be present without its normal content, provided that the nillable attribute is set to 'true' or 1. By default, the minoccurs and maxoccurs attributes are set to 1.</p> <p>In the following example, name is mandatory, age is optional, and phone is nillable.</p> <pre><s:element minOccurs="1" name="name" type="s:string" /> <s:element minOccurs="0" name="age" type="s:int" /> <s:element minOccurs="1" name="phone" nillable="true" type="s:string" /></pre> <p>Option availability per parameter type</p> <p>Mandatory</p> <ul style="list-style-type: none"> • Nil radio button: disabled • Include arguments in call: disabled <p>MinOccurs=0</p> <ul style="list-style-type: none"> • Nil radio button: disabled • Include arguments in call: enabled <p>Nillable</p> <ul style="list-style-type: none"> • Nil radio button: enabled • Include arguments in call: disabled <p>To include a specific optional argument in the service call:</p> <p>Click the node and select Include Argument in Call. The nodes for all included arguments are colored in blue. Arguments that are not included are colored in gray.</p> <p>If you include an element on a parent level, it automatically includes all mandatory and nillable children elements beneath it. If it is a child element, then it automatically includes the parent element and all other mandatory or nillable elements on that level. If you specify Generate auto-value to a parent element, VuGen provides values for those child elements that are included beneath the parent.</p>

Argument Type	Description
	<p>Note: VuGen interprets whether elements are mandatory or optional through the toolkit implementation. This may not always be consistent with the element's attributes in the WSDL file.</p>
Choice Optional Elements	<p>In a WSDL, defines a set of elements where only one of them appears in the SOAP message. In some cases, one of the Choice elements is optional, while the others are not. You can select the Choice element and still prevent its optional element from appearing in the SOAP envelope. In the Step Navigator, select the Choice element, and clear the Include argument in call option. In Script view, delete the line that defines the Choice argument.</p>
Recursive Elements	<p>Using the Properties dialog box, you can control the level of recursive elements to include in the Web Service call.</p> <p>To exclude a certain level and exclude those below, select the lowest parent node that you want to include and select Include Argument in Call. VuGen includes the selected nodes, its mandatory children, and all of its parent nodes.</p> <p>In the following example, three levels of the Choice argument are included—the rest are not. Excluded nodes are grayed out.</p> 

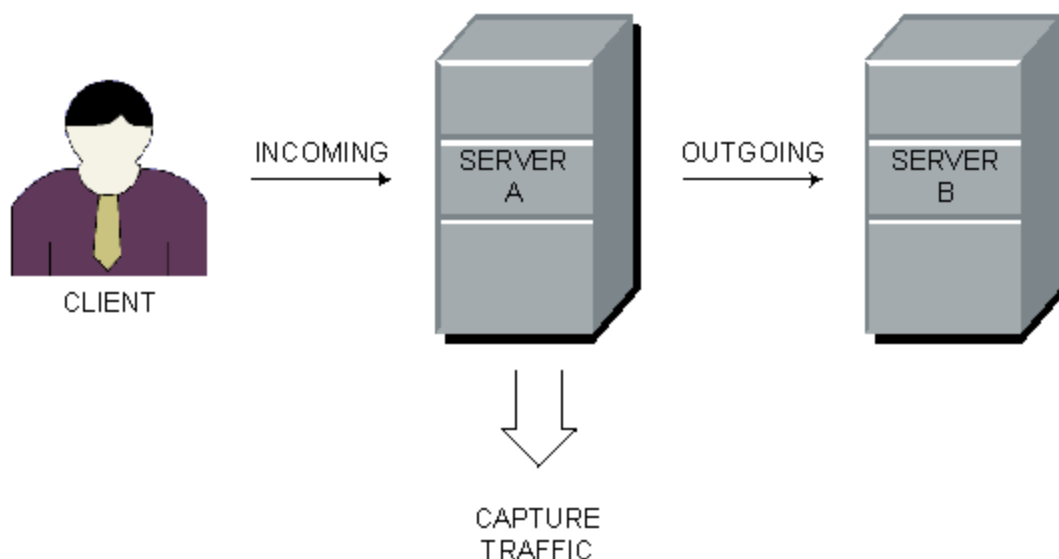
Argument Type	Description
Base64 Arguments	<p>An encoding method used to represent binary data as ASCII text. Since SOAP envelopes are plain text, you can use this encoding to represent binary data as text within SOAP envelopes.</p> <p>When VuGen detects a WSDL element of base64Binary type, it lets you provide an encoded value. You can specify a value in two ways:</p> <ul style="list-style-type: none">• Get from file. Reference a file name.• Embed encoded text. Specify the text to encode. <p>For details, see "Process Base64 Data - Simple Data Dialog Box" on page 748.</p>

Server Traffic Scripts Overview

The main focus when testing enterprises and complex systems, is to measure the performance from the client end. Ordinarily, VuGen records the actions you perform in the application or browser, and generates a script emulating the client actions and requests to the server.

In certain test environments, you may be unable to record the client application to retrieve the requests to the server. This may be a result of the server acting as a client, or because you do not have access to the client application. In these cases, you can create a script using VuGen's **Analyze Traffic** feature.

The **Analyze Traffic** feature examines a capture file containing the server network traffic, and creates a script that emulates requests sent to or from the server. The steps in creating a script by analyzing server traffic are described in ["Create a Script by Analyzing Traffic \(Web Services\)"](#) on page 737.



There are two types of emulations: **Incoming traffic** and **Outgoing traffic**.

Incoming traffic scripts emulate situations in which you want to send requests to the server, but you do not have access to the client application, for example, due to security constraints. The most accurate solution in this case is to generate a script from the traffic going **into** the server, from the side of the client.

When you specify an Incoming server network traffic, you indicate the IP address of the server and the port number upon which the application is running. VuGen examines all of the traffic going into the server, extracts the relevant messages, and creates a script. In the above diagram, if the client is unavailable, you could create an Incoming script to emulate the requests coming into **Server A**.

Outgoing Traffic scripts emulate the server acting as a client for another server. In an application server that contains several internal servers, you may want to emulate communication between server machines, for example between **Server A** and **Server B** in the above diagram. The solution in this case is to generate a script from the traffic sent as output **from** a particular server.

When you create an Outgoing traffic script, you indicate the IP address of the server whose outgoing traffic you want to emulate, and VuGen extracts the traffic going out of that server. In the above diagram, an Outgoing script could emulate the requests that **Server A** submits to the **Server B**.

- ["Create a PCAP File" on page 831](#)
- ["Filtering Traffic" below](#)
- ["Data on Secure Servers" on the next page](#)

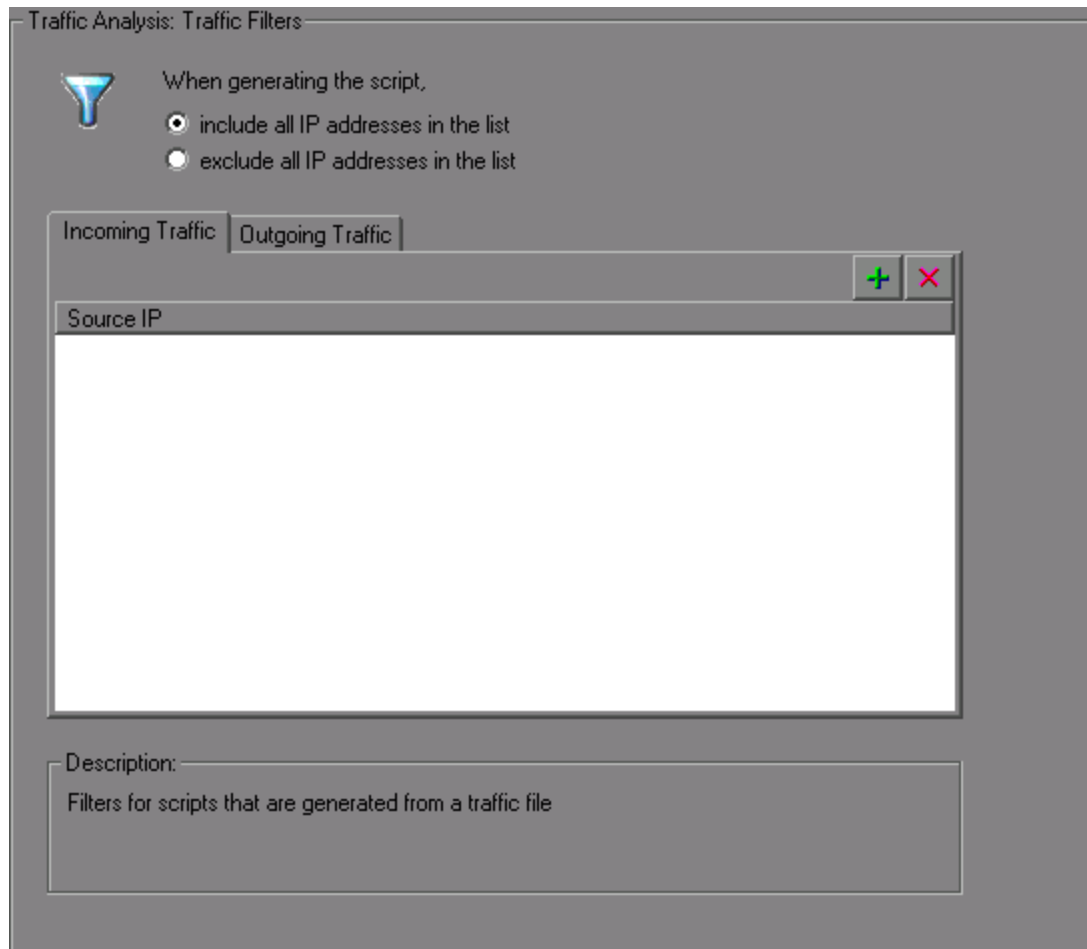
Filtering Traffic

You can provide a filter to drill down on specific requests going to or from a server, by specifying its IP address and port.



Tip: Several external capture tools allow you to filter the IP addresses while capturing the traffic.

You filter the requests by choosing the relevant host IP addresses. The filter can be inclusive or exclusive—you can include only those IPs in the list, or include everything except for those IPs that appear in the list.



For more information, see ["Create a Script by Analyzing Traffic \(Web Services\)"](#) on page 737.

Data on Secure Servers

To analyze traffic from a secure server, you must provide a certificate containing the private key of the server.

If the traffic is SSL encrypted, you must supply a certificate file and password for decryption. If you want traffic from multiple servers to be reflected in the script, you must supply a separate certificate and password for each IP address that uses SSL.

For more information, see ["Create a Script by Analyzing Traffic \(Web Services\)"](#) on page 737.

Add Content


This topic describes how to add content, such as Web Service calls, to a Web Services Vuser script.

Prerequisites

Create an empty Web Services Vuser script. Click **File > New Script and Solution** and choose the **Web**

Services protocol. You can create either a single-protocol or multi-protocol Vuser script.

Record a Session - Optional

1. Click the **Start Recording** button  on the VuGen toolbar or press Ctrl+R to open the Recording Wizard > Specify Services screen.
For user interface details, see ["Specify Services Screen" on page 737](#).
2. Add services to the list. Click **Import** to load a WSDL for the test. Indicate the location of the WSDL file.
For user interface details, see ["Import Service Dialog Box" on page 784](#).
3. Click **Next**. Specify the location of the application and any other relevant arguments. See the ["Specify Application to Record Dialog Box" on page 738](#).

Add a New Service Call - Optional

1. Import a service. Click **Manage Services** to access the Import dialog box.
For user interface details, see ["Import Service Dialog Box" on page 784](#).
2. Click the cursor at the desired location in your script (**Editor**) or in the test steps (**Step Navigator**).
3. Click the **Add Service Call** button. The New Web Service Call dialog box opens.
4. In the Select Web Service Call section, select a **Service**, **Port Name**, and **Operation**.
5. To specify an endpoint other than the default **Target Address**, select **Override Address** and insert the new endpoint to which you want to submit the requests.
6. Expand the nodes and specify argument values. To create sample values for all Input arguments, select the **Input Arguments** node and click **Generate**.



Tip: To instruct VuGen to interpret a string as text, and not markup data, you can use a CDATA section. For example, suppose you want to specify an input string "<3558>&abc". Instead of using escape characters such as `<3558>&abc`, add a CDATA section in the following format: `<![CDATA[<3558>&abc]]>`.

To edit, import, or export the element's XML structure, see ["Assign Values to XML Elements" on the next page](#).

7. To parameterize an input argument, click the node and select the **Value** option. Click the ABC icon and proceed with parameterization. For more information, see ["Parameterization" on page 342](#).
8. Select the **Transport Layer Configuration** node to specify advanced options, such as JMS transport for SOAP messages (Axis toolkit only), asynchronous messaging, or WS Addressing. For more information, see ["Send Messages over JMS" on page 764](#).

Add Attachments - Optional

1. To add an attachment to an input argument, choose an operation in the left pane. Select **Add to request (Input)**. VuGen prompts you to enter information about the attachment and adds it to the

method's tree structure. For details, see the ["Add Input Attachment Dialog Box" on page 747](#).

2. To specify an output attachment through which to store output arguments, choose an operation in the left pane. Select **Save received (Output)**. Select the desired option: **Save All Attachments** or **Save Attachment by Index** based on their index number—beginning with 1. For details, see ["Web Service Call Attachments" on page 727](#).
3. To edit the properties of either an Input or Output attachment, click the attachment in the left pane, and enter the required information in the right pane.

Specify SOAP Headers - Optional

Select the **CustomSOAP Header** node in the left pane and enable the **Use SOAP header** option. You must individually specify SOAP headers for each element. To compose your own, click **Edit** and edit the XML. To import an XML file for the SOAP header, click **Import**.

Import SOAP - Optional

1. Import a service if one is available. Click **Manage Services** to access the Import dialog box. For more information, see the ["Import Service Dialog Box" on page 784](#).
2. Click the **Import SOAP** button to open the Import SOAP dialog box.
3. Browse for the XML file that represents your SOAP request.
4. Select the type of step you would like to generate: **Create Web Service Call** or **Create SOAP Request**. In order to create a Web Service Call, you must first import at least one WSDL that describes the operation in the SOAP request file. To view the SOAP before loading it, click **View SOAP**.
5. Click **Load** to import the XML element values.
For a **Web Service Call**, set the properties for the Service call as described in the ["New Web Service Call Dialog Box" on page 740](#).
For a **SOAP Request**, provide the URL and the other relevant parameters.
6. For a Web Service Call, if there are multiple services with same operation (method) names, select the service whose SOAP traffic you want to import.
7. Click **OK** to generate the new step within your script.

Analyze Server Traffic - Optional

To create a script by analyzing a file containing a dump of the server traffic, click **Analyze Traffic**.

For details, see ["Server Traffic Scripts Overview" on page 731](#).

Assign Values to XML Elements

This task describes how to work with XML elements by manually editing the code, importing an external file, and exporting the XML for later use.

1. Prerequisite
Import a service and create a new Web Service call. Alternatively, in the Step Navigator, right-click a step and select **Show Arguments**.

2. Select the element.

In the left pane, select a complex type or array argument. In the right pane, click **XML**. The XML field shows the XML code as a single string.

3. Import a file (optional).

To import a previously saved XML file, click **Import** and specify the file's location.

4. Edit the XML elements (optional).

To edit the XML structure and element values, click **Edit**. The XML Editor opens. To import a previously saved XML file, click **Import File**.

- To manually edit the code, click the **Text View** tab.
- To modify the XML through a graphical interface, click the **Step Navigator**. Use the shortcut menu to add children and sibling elements and rename the node. Click **Insert** from the shortcut menu to add a new element before or after the selected one.



Tip: To instruct VuGen to interpret a string as text, and not markup data, you can use a CDATA section. For example, suppose you want to specify an input string "<3558>&abc". Instead of using escape characters such as `<3558>&abc`, add a CDATA section in the following format: `<![CDATA[<3558>&abc]]>`.

5. Export a file (optional).

To save your XML data to a file so it can be used for other tests, click **Export** and specify a location.

Generate a Test Automatically

This task describes how to create requirements or tests for checking your service.

1. **Open the wizard**

Select **File > New** to open the New Virtual User dialog box. Select **SOA Test Generator** in the left pane and click **Create**.

2. **Add a service**

Proceed to the next screen and click **Add** to import at least one service. If your service is not ready yet, you can use an emulated service. For details, see ["Add and Manage Services" on page 779](#). Click **Next**.

3. **Select testing aspects**

Expand the nodes and select the desired testing aspects. Click **Next**.

4. **Specify a location**

Specify a test name and a location for the test scripts: **HPE ALM** or a **local file system**. If you specified ALM, click **Connect** to log on to the server and **Browse** to locate the test node.

5. **Complete the test generation**

Review the summary and include or exclude any scripts from the generation. Click **Generate**.

6. Open the scripts

In the final screen, review the list of generated scripts and indicate which ones to open. Click **Finish**.

Create a Script by Analyzing Traffic (Web Services)

This task describes how to create a Web Services Vuser script using a network traffic file. You can use a capture file of the following types: our **pcap**, **pcap**, **lrcap**, or **saz** (Fiddler)




1. Prerequisite: Create a capture file on a Windows platform. (Optional)
Locate your capture file or create a new one using an external tool, such as Wireshark. For details on creating a capture file, see ["Create a PCAP File" on page 831](#).
2. Open the Analyzing Traffic wizard.
Create a new Web Services Vuser script and click **Analyze Traffic**. The ["Specify Services Screen"](#) opens.
3. Import a service and add it to the list. (Optional)
 - a. Click **Import** to load a WSDL file. For details, see the ["Import Service Dialog Box" on page 784](#).
 - b. Click **Next**.
4. Specify traffic information.
 - a. Browse for the capture file.
 - b. Indicate whether you want to analyze **Incoming** or **Outgoing** traffic and specify the server whose traffic you want to analyze.
 - c. Select the section of the script into which you want to load the traffic: **vuser_init**, **Action**, or **vuser_end**.
5. Filter the IP addresses. (Optional)
Click the **Filter Options** button to open the Recording options and indicate which IP addresses to ignore or include. For details, see ["Traffic Analysis > Traffic Filters Recording Options" on page 202](#).
6. Configure the SSL. (Optional)
Click the **SSL Configuration** button to add SSL certificates. This is necessary in order to analyze traffic from a secure server. For details, see the ["SSL Configuration Dialog Box" on page 752](#).

Specify Services Screen

This dialog box enables specify the basic details needed to begin recording a Web Services script.

To access	VuGen > Start Record button
Relevant tasks	"Add Content" on page 733

User interface elements are described below (unlabeled elements are shown in angle brackets):

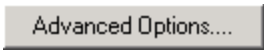

UI Element	Description
	Opens the Manage Services dialog box for providing further information about the service. For more information, see "Manage Services Dialog Box" on page 781 .
	Opens the Import Service dialog box. For more information, see the "Import Service Dialog Box" on page 784 . For user interface details, see the "Import Service Dialog Box" on page 784 .
	Removes the selected service from the list.
<services list>	<p>A list of the available services:</p> <ul style="list-style-type: none"> • Service Name. The native name of the service. • WSDL Location. The source of the WSDL.

Specify Application to Record Dialog Box

This dialog box enables specify the basic details needed to begin recording a Web Services script.

To access	VuGen > Start Record button, Next
Relevant tasks	"Add Content" on page 733


User interface elements are described below:

UI Element	Description
	Opens the Recording Options dialog box. For user interface details, see "Recording Options" on page 141 .
Record default Web browser	<p>Records the default browser actions. Specify the starting URL or click the Browse button to navigate to a location.</p> <div>  Note: The Web Services protocol only supports IE as the default browser. </div>


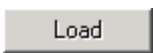
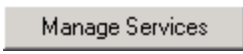
Record any application	<p>Records any Win32 application. You can also specify the following:</p> <ul style="list-style-type: none"> • Program to record. Select the browser, internet application, or Win32 application to record • Program arguments (Win32 Applications only). Command line arguments for the application. For example, if you specify plus32.exe with the command line options peter@neptune, it connects the user Peter to the server Neptune when starting plus32.exe. • Working directory. A working folder for the application (only when required by the application).
Record into action	<p>The section into which you want to record: vuser_init, Action, or vuser_end. For actions you want to repeat, use the Action section. For initialization steps, use vuser_init.</p>
Record application startup	<p>In the following instances, it may not be advisable to record the startup:</p> <ul style="list-style-type: none"> • If you are recording multiple actions, in which case you need to perform the startup in only one action. • In cases where you want to navigate to a specific point in the application before starting to record. • If you are recording into an existing script.


Import SOAP Dialog Box

This dialog box enables you to create a test step based on a SOAP file.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> • Click  Import SOAP • SOA Tools > Import SOAP
Relevant tasks	<p>"Add Content" on page 733 "Adding Web Service Script Content - Overview" on page 725</p>


User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Browse. Locate the XML file containing SOAP traffic.
	Loads the element values from the SOAP file.
	Opens the Manage Services dialog box for importing and configuring services.

	Opens the SOAP file in a browser for viewing.
<Call type>	The type of call to generate in the script/test: <ul style="list-style-type: none"> • Web Service Call. Requires the import of a service. • SOAP Request. Generates a soap_request step in the script.
SOAP Request Properties	The properties of the SOAP request (only visible for SOAP Request type calls). Specify the following: <ul style="list-style-type: none"> • URL. The URL or IP address of the server to which to submit the request. • SOAP Action. The SOAP action to include in the request (applicable if there are multiple actions). • Response Parameter. A parameter name to store the response of the SOAP or Web Service call request.

New Web Service Call Dialog Box

This dialog box lets you create and configure a new Web Service call.

To access	Open a Web Service Vuser script and then click SOA Tools > Add Service Call or click the Add Service Call button  on the VuGen toolbar.
Important Information	To access the Web Service call properties for existing Web Service calls, select a step in the Step Navigator and choose Properties from the shortcut menu.
Relevant tasks	"Add Content" on page 733

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<service argument tree> (left pane)	An expandable tree hierarchy of the Service containing the following nodes: <ul style="list-style-type: none"> • <operation name> • Transport Layer Configuration • Custom SOAP Header • Input Arguments • Output Arguments
<parameter values> (right pane)	Enables you to set and select values for each of the left pane's nodes.

Select Web Service Call	<p>Lets you set the following items:</p> <ul style="list-style-type: none"> • Service. A drop-down list of all of the imported services, with the name derived from the WSDL. • Port Name. A drop-down list of available ports through which to send the request. • Operation. A drop-down list of the service's operations. • Target Address. The default endpoint of the service. • Override Address. Allows you to enter an alternate endpoint address in the Target Address box.
--------------------------------	---

<Operation Name> Node

Allows you to generate sample values for the operation's input arguments and add attachments.



User interface elements are described below:

UI Element	Description
Method	The name of the selected operation (read-only).
Description	Free text area for entering a description of the service.
Attachments	<p>Handles input and output attachments:</p> <ul style="list-style-type: none"> • Add to Request (Input). Attaches a file or parameter value to the request. • Save Received (Output). Saves the response to a parameter.
Step properties	<p>Lists the following service call properties and their values:</p> <ul style="list-style-type: none"> • WSDL file location • Service • Port name • Target address • SOAP action • SOAP namespace

Transport Layer Configuration Node

User interface elements are described below:

UI Element	Description
------------	-------------

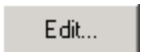

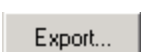

HTTP/S Transport	Sets the transport method to HTTP or HTTPS transport.
Async Support	<p>Marks the Web Service call as an asynchronous message activated by an event: Async Event. An arbitrary name for the event.</p> <p>Note: Add a Web Service Wait For Event step to the script, to instruct the replay engine to wait for the event.</p>
WSA Support	<p>Enables WS-Addressing. Use one of the following options for a reply:</p> <ul style="list-style-type: none"> • WS-A Reply. An IP address of the server to reply to when the event occurs. • Autodetect. Reply to the current host when the event occurs. This is useful when running the same script on several machines. <div>  <p>Tip: To use WS-Addressing calls in synchronous mode, leave the Async Event box empty. In Script view, remove the AsyncEvent argument. This instructs the replay to block script execution until the complete response is received from the server.</p> </div>
JMS Transport	<p>Sets the transport method to JMS for <i>synchronous</i> messages. For details, see "Testing Web Service Transport Layers Overview" on page 754.</p> <div>  <p>Note: For JMS <i>asynchronous</i> messages, manually add a JMS Send Message Queue or JMS Receive Message Queue step to the script, to set up the message queue information.</p> </div>
Override JMS Queues	Enables you to provide the request and response queues.
Request Queue	The queue name for the request message.
Response Queue	The queue name for the response message.

Custom SOAP Header Node

Lets you specify additional application-generated header elements to include in the SOAP envelope of an HTTP message. For task details, see ["Add Content" on page 733](#).

User interface elements are described below:



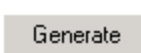
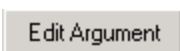


UI Element	Description
------------	-------------

	Opens an XML editor that lets you view and edit the SOAP header XML code.
	Opens the Select XML File to Import dialog box.
	Opens the Export SOAP Header into File dialog box.
	Opens the Select or Create New Parameter dialog box.
Use SOAP Header	Includes a SOAP header in the HTTP request.
Header	The header source: <ul style="list-style-type: none"> • For an imported file: Header element as it appears in the imported file. • For a parameter: The parameter name (in curly brackets)

Input Arguments Node

Lets you set the properties and generate values for all input arguments.

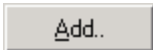



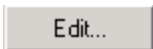



User interface elements are described below:

UI Element	Description
	Includes all of the method's arguments in the Web Service call.
	Resets the arguments to their original state. It removes their inclusion in the call, and sets them to the values in the WSDL.
	Generates sample data for all of the input arguments.
	Opens the pane for editing the selected argument's value.
Name	The name of the operation (read-only)
Argument List	A list of the input arguments. <ul style="list-style-type: none"> •  Simple parameters •  Arrays (shows the top level only).

<Input Argument Name> Node

When selecting an input argument, the right pane allows you to specify argument values.

User interface elements are described below:

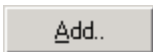
UI Element	Description
	Opens the "Add Array Elements Dialog Box" on page 748 for adding a new array element to the input argument (only visible when selecting a parent node in an input array).
	Removes the selected array element in the input argument (only visible when selecting a parent node in an input array).
	Includes the sub-arguments of the selected argument, in the Web Service call. This is only enabled for an argument with sub-arguments with the Include argument in call option enabled.
	Excludes the sub-arguments of the parent argument, from the Web Service call.
	Opens an XML editor for editing the XML code containing the argument values. The only changes saved are the element values and the number of array elements.
	Opens the Select XML File to Import dialog box,
	Opens the Export argument XML into file dialog box.
	Opens the Select or Create Parameter dialog box.
Name	The name of the argument or array.
Include argument in call	Include the argument in the call. For arrays, click Include to add the sub arguments to the call. To exclude all omissible arguments, click Exclude .
Type	The argument type as defined in the WSDL. When the WSDL contains derived types, this box becomes a drop-down list. For details, see "Special Argument Types" on page 728 .
Nil	Sets the Nillable attribute to true .
XML (for arrays only)	<ul style="list-style-type: none"> • XML. Enables the Edit, Import, and Export buttons. By editing the XML, you can manually insert argument values. Click on the ABC icon to replace the entire XML structure with a single XML type parameter. Note: This import operation handles XML files that were previously exported—not standard SOAP files. • Generate auto-value for this argument. Inserts automatic values for all children elements. • Add/Delete. Adds or removes elements from the array.

Value (for non -array elements)	The argument value. To parameterize this value, click the abc icon (only available for non-arrays).
Generate auto-value for this argument	Generates a sample value for the selected argument.

Input Attachments Node

Lets you add attachment to the input arguments. This is only visible if you enabled **Add to request (Input)** in the operation name (top level parent) node.


User interface elements are described below:

UI Element	Description
	Open the Add Input Dialog box. In this dialog box, you specify the source, content-type, and content-ID of the attachment.
Attachment format	The format of the attachment, such as MIME, DIME , and so forth.

<Input Argument Attachment> Node

Lets you set the properties for input attachments. This is only visible if you enabled **Add to request (Input)** in the operation name (top level parent) node.


User interface elements are described below:

UI Element	Description
	Deletes the selected attachment parameter. If you have saved the attachments by index, it only removes the selected item.
Take data from	The source of the attachment—either a file, or an existing parameter.
Content-type	The content type of the parameter: Detect automatically or specify a value such as application/octet-stream .
Content-ID	A unique content ID for the parameter: Generate automatically or specify a value.

Output Arguments Node

Lets you see the properties of all output arguments.



User interface elements are described below:

UI Element	Description
	Opens the pane for editing the selected argument's value.
Name	The name of the operation (read-only).
Argument List	A list of the output arguments and the corresponding parameters storing the values.

<Output Argument Name> Node

Lets you specify a parameter for storing the value of the output argument.

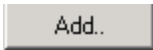
User interface elements are described below:

UI Element	Description
	Opens the "Add Array Elements Dialog Box" on page 748 for adding a new array element to the output argument (only visible when selecting a parent node in an output array). For details, see the "Add Array Elements Dialog Box" on page 748 .
	Removes the selected array element in the output argument (only visible when selecting a parent node in the output array).
Name	The name of the output argument or array.
Save returned value in parameter	Saves the value of the selected argument to a parameter. To specify a custom parameter name, modify the default Param_<arg_name> in the Parameter field.
Nil	Sets the value of the current argument to nil=true .
XML (for arrays only)	XML code containing the argument values. To parameterize this value, click the abc icon (only available for arrays).

Output Attachments Node

Lets you set the properties for output attachment parameters. This is only visible if you enabled **Save received (Output)** attachments in the operation name (top level parent) node.

User interface elements are described below:


UI Element	Description
	Adds a new index-based output argument. This is available only when choosing Save Attachments by Index .

Save All Attachments	Saves all output attachments to a parameter with the following properties: <ul style="list-style-type: none"> • Content. An editable name for the parameter storing the attachment • Content-type. The type of parameter (read-only). • Content-ID. A unique ID for the parameter (read-only).
Save Attachments by Index	Saves the output attachments to index-based parameters. To set the index, select one of the parameters and modify the index number in the right pane.

<Output Argument Attachment> Node


Lets you set the properties for output attachments. This is only visible if you enabled **Save received (Output)** attachments in the operation name (top level parent) node.

User interface elements are described below:

UI Element	Description
	Deletes the selected attachment parameter. If you have saved the attachments by index, it only removes the selected item.
Index	An index number for the parameter. This field is only enabled when you select Save Attachments by Index in the Output Attachments node.
Content	An editable name for the parameter storing the attachment
Content-type	The content type of parameter (read-only).
Content-ID	A unique content ID for the parameter (read-only).

Add Input Attachment Dialog Box

This page enables you to add input attachments to your Web requests.

To access	Click  Add Service Call and select the top node—the Operation name. Select Add to Request (input) in the Attachments section.
Important information	You must import a service before adding an attachment to a service call. For background information, see "Web Service Call Attachments" on page 727 .
Relevant tasks	"Add Content" on page 733

User interface elements are described below:


UI Element	Description
------------	-------------

Take data from	<p>The location of the data.</p> <ul style="list-style-type: none"> • File. The file location: <ul style="list-style-type: none"> • Absolute Path: The full path of the file. Note that this file must be accessible from all machines running the script. • Relative Path: (recommended) A file name. Using this method, during replay, VuGen searches for the attachment file in the script's folder. To add it to the script's folder, select File > Add Files to Script and specify the file name. • Parameter. The name of a parameter containing the data.
Content-type	The content type of the file containing the data. The Detect Automatically option instructs VuGen to automatically determine the content type. The Value box accepts manual entries and provides a drop-down list of common content types.
Content-id	A unique identifier for the attachment. By default, VuGen generates this automatically. Optionally, you can specify another ID in the Value box.

Add Array Elements Dialog Box

This page enables you to add elements to an argument array with an identical structure to the existing array. This is available for both Input and Output arguments.

For Input elements, you can base the new array's values on an existing element.

To access	Click  Add Service Call . Select an argument node that is an array.
Important information	You must have an array in your argument tree in order to view this dialog box.
Relevant tasks	"Add Content" on page 733

User interface elements are described below:


UI Element	Description
Name	The name and index of the array's parent node.
Start Index	The index from which to add new array elements.
Elements	The number of identical array elements to add to your argument tree.
Copy values from index	Creates the new array elements with the values of a specific array element (only available for Input arguments).

Process Base64 Data - Simple Data Dialog Box

This dialog box enables you to set the encoding options for your simple base64 data.

To access	<p>For simple, non-complex Base64 values:</p> <ul style="list-style-type: none"> • Select an input argument in the Web Service Call Properties of Base64 type. • Select the Value option. • Choose Embed encoded text. • Click the Browse button.
Important information	For a complex array, use the " Process Base64 Data - Complex Data Dialog Box " below.
Relevant tasks	"Add Content" on page 733

User interface elements are described below:


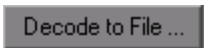

UI Element	Description
	Allows you to save the decoded text to a file.
Text to encode	For complex data, use the " Process Base64 Data - Complex Data Dialog Box " below.
Encoding Options	A list of encoding methods. Default: Unicode (UTF-8)
Encoded data	The encoded version of the data from the Text to encode pane.

Process Base64 Data - Complex Data Dialog Box

This dialog box enables you to set the encoding options for your complex base64 data.

To access	<p>For complex Base64 values:</p> <ul style="list-style-type: none"> • Select a complex input argument in the Web Service Call Properties, of Base64 type. • Select the Value option click the Parameter icon. • Replace the value with a parameter. • Right-click the Parameter icon in the Value box and select Parameter Properties. • Click the Edit Data button. • In the desired values set column, click the B64 button.
Important information	For simple, non-complex data, use the " Process Base64 Data - Simple Data Dialog Box " on the previous page.
Relevant tasks	"Add Content" on page 733

User interface elements are described below:

UI Element	Description
	Encodes the specified file.
	Enables you to save decoded data to a file. This is usually data obtained during replay.
File	<p>Encode the file by reference or its contents.</p> <ul style="list-style-type: none"> • File path. The file to encode. • Link to file. References the file containing the values. If cleared, it uses the content of the specified file. It copies the content to the script folder. <div>  Tip: For text exceeding 10KB, enable Link to file. </div>
Text	<p>Encodes the specified text string.</p> <ul style="list-style-type: none"> • Text to encode. The Base64 text to encode. As you type the text, VuGen encodes it in the Encoded data pane. • Encoding Options. A list of encoding methods. The default is Unicode (UTF-8).

Aspects List

The following table lists the available testing aspects:

Aspect Name	Description
Positive Testing	Generates a full positive test that checks each operation of the service.
Standard Compliance	Checks the service's compliancy with industry standards such as WS-I and SOAP.
Service Interoperability	<p>Tests the interoperability of the service's operations with all supported Web Services toolkits.</p> <p>Contains the following sub-aspects:</p> <ul style="list-style-type: none"> • .NET Framework. Tests that the services are fully interoperable with .NET Framework WSE 2 Toolkit by calling all of its operations with default/expected values. • Axis/Java Based Web Services. Tests that the services are fully interoperable with Axis 1.3 Web Services Framework by calling all of its operations with default/expected values.

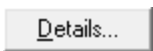

Security Testing	<p>Tests service security. Contains the following sub-aspects:</p> <ul style="list-style-type: none"> • SQL Injection Vulnerability. Checks if the service is vulnerable to SQL injections by injecting SQL statements and errors into relevant parameters. • Cross-site Scripting (XSS). Attempts to hack the service by injecting code into a Web site that will disrupt its functionality.
Boundary Testing	<p>Using the negative testing technique, creates tests to manipulate data, types, parameters, and the actual SOAP message to test the service to its limits. Contains the following sub-aspects:</p> <ul style="list-style-type: none"> • Extreme Values. Provides invalid data types to the services and verifies they are not accepted. • Null Values. Provides NULL parameters to the services to verify they are not accepted.
Performance Testing	<p>Contains the following sub-aspects:</p> <ul style="list-style-type: none"> • Stress Testing. Tests the maximum load that can be placed on the application. • Overload Sustainability Testing. Tests how well the hardware allocated for the application can support the number of anticipated users. • Volume Testing. Tests that the system can handle a massive data entry. • Longevity Test. Tests that the system can sustain a consistent number of concurrent Vusers executing transactions using near-peak capacity, over a minimum 24-hour period. • Scalability Testing. Repeated stress, overload, volume, and longevity tests with different server or network hardware configurations.


Specify Services Screen

This wizard screen enables you to select Web services to associate with your traffic-based script.

To access	Analyze Traffic button
Relevant tasks	"Add Content" on page 733

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Opens the Manage Services dialog box for providing further information about the service. For more information, see "Manage Services Dialog Box" on page 781 .
	Opens the Import Service dialog box. For more information, see "Import Service Dialog Box" on page 784 .


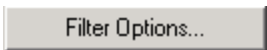

	Removes the selected service from the list.
<services list>	<p>A list of the available services:</p> <ul style="list-style-type: none"> • Service Name. The native name of the service. • WSDL Location. The source of the WSDL.

Specify Traffic Information Screen

This wizard screen enables you to specify the capture file for the incoming or outgoing traffic.


To access	Analyze Traffic button, Next
Relevant tasks	"Add Content" on page 733

User interface elements are described below:




UI Element	Description
	Browse. Allows you to select a capture file to import.
	Opens the Traffic Filters node in the Recording Options dialog box. This allows you to specify which IP addresses to include or exclude from the script. For details, see "Recording Options" on page 141 .
	Opens the "SSL Configuration Dialog Box" below which allows you to add SSL certificates to analyze traffic from a secure server.
Capture file	The name of a capture file containing the server traffic, usually with a cap extension.
Incoming Traffic	The IP address and port of the server whose incoming traffic you want to examine.
Outgoing Traffic	The IP address of the server whose outgoing traffic you want to examine.
Record into action	The section into which you want to create the script: vuser_init , Action , or vuser_end . For actions you want to repeat, use the Action section. For initialization steps, use vuser_init .

SSL Configuration Dialog Box

This dialog box enables you to configure the SSL information for decrypting your traffic file.

To access	Analyze Traffic button, Next , 
Relevant tasks	"Add Content" on page 733

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Add Certificate. Adds a new entry to the certificate list, containing information about an SSL server and its private key.
	Delete. Removes the selected private key.
<certificate list>	<p>The properties of the SSL entry. Specify the following:</p> <ul style="list-style-type: none"> • IP. IP address of the server being analyzed. • Port. Port of the server being analyzed. • File. The path of the certificate file (with a pem extension) containing the private key. Use the Browse button to locate the file. • Password. A password for decrypting the private key. <p>See Certificate Utility below for details on how to convert certificates.</p>
	<p>Opens the Convert Certificate tab of the SSL Utility that enables you to convert certificates from PKCS #12 and X.509 formats to PEM format.</p> <p>For details on how to convert the certificate to PEM format, see web_set_certificate_ex in the Function Reference (Help > Function Reference).</p>

Web Services - Preparing Scripts for Replay

Preparing for Replay Overview

After you create a script with Web Service calls, you prepare it for replay.

You can enhance it with custom error and log messages or with transactions. In addition, you can enhance your script with JMS functions, **jms_<suffix>** or XML functions, **lr_xml_<suffix>**. For more information, see the Function Reference (**Help > Function Reference**).

Runtime settings let you emulate real users more accurately, configure the runtime settings. These settings include general settings and Web Service specific settings. For details, see ["Runtime Settings Overview" on page 283](#).

In certain cases, you may need to use the result of one Web Service call as input for another. To do this, you save the result to a parameter and reference it later. For more information, see ["Prepare Scripts for Replay" on page 763](#).

Testing Web Service Transport Layers Overview

Web services can be sent over various transport layers. The transport layer is the protocol used to transport messages to and from the server.

VuGen allows you to configure the transport layer for your services. It fully supports HTTP/HTTPS and JMS (Java Message Service) transport layers.

With user handlers, you can process SOAP requests and responses and assign them a custom behavior. For more information, see ["User Handlers" on page 758](#).

- ["Sending Messages over HTTP/HTTPS" below](#)

Sending Messages over HTTP/HTTPS

HTTP is used for sending requests from a Web client, usually a browser, to a Web server. HTTP is also used to return the Web content from the server back to the client.

HTTPS handles secure communication between a client and server. Typically, it handles credit card transactions and other sensitive data.

The typical request and response mechanism is synchronous. In synchronous messaging, the replay engine blocks script execution until the server sends its response. In asynchronous mode, the replay engine executes the script without waiting for server's response for previous messages.

If you are working with HTTP or HTTPS transport, you can use asynchronous calls in conjunction with WS-Addressing. For details, see ["WS-Addressing" on page 757](#).

JMS Transport Overview

JMS is a J2EE standard for sending messages, either text or Java objects, between Java clients.

Communication scenarios:

- **Peer-to-Peer.** Also known as **Point-to-Point**. JMS implements point-to-point messaging by defining a message queue as the target for a message. Multiple senders send messages to a message queue, and the receiver gets the message from the queue.
- **Publish-Subscribe.** Each message is sent from one publisher to many subscribers through a designated topic. The subscribers only receive messages sent after they have subscribed.

VuGen supports point-to-point communication by allowing you to send and receive JMS messages to and from a queue.

Before you can send messages over JMS transport, you need to configure several items that describe the transport:

- **JNDI initial context factory.** The class name of the factory class that creates an initial context which will be used to locate the JMS resources such as JMS connection factory or JMS queue.
- **JNDI provider.** The URL of the service provider which will be used to locate the JMS resources such as JMS connection factory or JMS queue.
- **JMS connection factory.** The JNDI name of the JMS connection factory.

In addition, you can set a timeout for received messages and the number of JMS connections per process.

You configure these settings through the JMS runtime settings. For details, see the **JMS > Advanced** view.

See also:

- ["JMS Script Functions" below](#)
- ["JMS Message Structure" on the next page](#)

JMS Script Functions

VuGen uses its API functions to implement the JMS transport. Each function begins with a **jms** prefix:

Function Name	Description
jms_publish_message_topic	Publishes messages to a specific topic
jms_receive_message_queue	Receives a message from a queue
jms_receive_message_topic	Receives published messages to a specific topic on a subscription.
jms_send_message_queue	Sends a message to a queue.
jms_send_receive_message_queue	Sends a message to a specified queue and receives a message from a specified queue.
jms_subscribe_topic	Creates a subscription for a topic.
jms_set_general_property	Sets a general property in the user context.
jms_set_message_property	Sets a JMS header or property for the next message to be sent, or uses a JMS header or property to filter received messages.

The JMS steps/functions are only available when manually creating scripts—you cannot record JMS messages sent between the client and server.

Unlike peer-to-peer communication that uses message queues, the publish-subscribe functions, **jms_publish_message_topic**, **jms_subscribe_topic**, and **jms_receive_message_topic**, are not supported for Web Service calls. To use these functions with Web Service calls, you must manually set up user handlers to generate the JMS message payload. For more information, see ["Create a User Handler" on page 768](#).

For details about the JMS functions, see the [Function Reference \(Help > Function Reference\)](#) (or click **F1** on the function).

JMS Message Structure

Each JMS message is composed of:

- **Header.** contains standard attributes (Correlation ID, Priority, Expiration date).
- **Properties.** custom attributes.
- **Body.** text or binary information.

JMS can be sent with several message body formats. Two common formats are **TextMessage** and **BytesMessage**.

To override the default behavior, use a **jms_set_general_property** function before sending the message. Set the JMS_MESSAGE_TYPE property to TextMessage, BytesMessage, or Default. For Example:

```
jms_set_general_property("step1","JMS_MESSAGE_TYPE","BytesMessage");
```

For more information, see the [Function Reference \(Help > Function Reference\)](#).

Asynchronous Messages Overview

You can use VuGen to emulate both synchronous and asynchronous messaging.

In synchronous messaging, the replay engine blocks script execution until the server sends its response. In asynchronous mode, the replay engine executes the script without waiting for server's response for previous messages.

This section also includes:

- ["Sending Asynchronous Calls with HTTP/HTTPS" below](#)
- ["WS-Addressing" on the next page](#)

Sending Asynchronous Calls with HTTP/HTTPS

This following section describes how to use asynchronous calls in HTTP/HTTPS. You use a **Wait for Event** step to instruct Vusers to wait for the response of previous asynchronous requests before continuing. The listener blocks the execution of the service until the server responds.

When adding a Web Service Wait for Event step, you specify the following:

- **Quantifier.** The quantifier indicates whether the Vuser should wait for **ALL** events to receive a response or **ANY**, just one of them. **ANY** returns the name of the first event to receive a response. **ALL** returns one of the event names.
- **Timeout.** the timeout in milliseconds. If no events receive responses in the specified timeout, then **web_service_wait_for_event** returns a NULL.
- **Events.** a list all of the asynchronous events for which you want to wait.

When running a script with asynchronous messaging, the Replay log provides information about the events and the input and output arguments.

For task details, see ["Send Messages over HTTP/S" on page 765](#).

When setting up an asynchronous message, you can set the location to which the service responds when it detects an event using WS-Addressing. For more information, see ["WS-Addressing" below](#).

WS-Addressing

WS-Addressing is a specification that allows Web Services to communicate addressing information. It does this by identifying Web service endpoints in order to secure end-to-end endpoint identification in messages. This allows you to transmit messages through networks that have additional processing nodes such as endpoint managers, firewalls, and gateways. WS-Addressing supports Web Services messages traveling over both synchronous or asynchronous transports.

The WS-Addressing specification requires a **WSAReplyTo** address—the location to which you want the service to reply.

An optional **WSAAction** argument allows you to define a SOAP action for instances where transport layers fails to send a message.

The following example illustrates a typical SOAP message using WS-Addressing, implemented in the background by VuGen.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://www.w3.org/2004/12/addressing">
  <S:Header>
    <wsa:MessageID>
      http://example.com/SomeUniqueMessageIdString
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://myClient.example/someClientUser</wsa:Address>
    </wsa:ReplyTo>
    <wsa:Address>http://myserver.example/DemoErrorHandler</wsa:Address>
    </wsa:FaultTo>
    <wsa:To>http://myserver.example/DemoServiceURI</wsa:To>
    <wsa:Action>http://myserver.example/DoAction</wsa:Action>
  </S:Header>
  <S:Body>
    <!-- Body of SOAP request message -->
  </S:Body>
</S:Envelope>
```

In the following example, the server responds to the interface 212.199.95.138 when it detects Event_1.

```
web_service_call("StepName=Add_101",
    "SOAPMethod=Calc.CalcSoap.Add",
    "ResponseParam=response",
    "AsyncEvent=Event_1",          "WSAReplyTo=212.199.95.138",
    "WSDL=http://lab1/WebServices/CalcWS/Calc.asmx?wsdl",
```

```
"UseWSDLCopy=1",  
"Snapshot=t1153825715.inf",  
BEGIN_ARGUMENTS,  
    "first=1",  
    "second=2",  
END_ARGUMENTS,  
BEGIN_RESULT,  
    "AddResult=Param_AddResult1",  
END_RESULT,  
LAST);
```

You can issue WS-Addressing calls in both asynchronous and synchronous modes. To use WS-Addressing in synchronous mode, leave the **Async Event** box empty in the Transport Layer options. In Script view, remove the **AsyncEvent** argument. This instructs the replay engine to block script execution until the complete response is received from the server.

For task details, see ["Send Messages over HTTP/S" on page 765](#).

Customizing Overview

VuGen provides several advanced capabilities that allow you to customize the way your script behaves. These capabilities are user handlers and configuration files.

With user handlers, you can process SOAP requests and responses and assign them a custom behavior. For more information, see below.

Configuration files let you customize advanced settings such as security information and the WSE configuration.

- ["User Handlers" below](#)
- ["Custom Configuration Files" on page 760](#)

User Handlers

User Handlers are open APIs through which you can perform the following operations:

- Get and set the request/response SOAP envelopes
- Override the transport layer
- Get and set the request/response content type
- Get and Set values for LoadRunner parameters
- Retrieve a configuration argument from the script
- Issue messages to the execution log
- Fail an execution

You can set up a user handler directly in a script, or implement it through a DLL. You can apply the handler locally or globally.

For task details, see ["Create a User Handler" on page 768](#).

For sample user handlers, see ["User Handler Examples" on page 760](#).

Handler Function Definitions

For basic implementation of a user handler, you define a user handler function within your Vuser script with the following syntax:

```
int MyScriptFunction(const char* pArgs, int isRequest)
```

The **pArgs** argument contains the string that is specified in **UserHandlerArgs** argument of **web_service_call** function.

The **isRequest** argument indicates whether the function is being called during processing of a Request (1) or Response (0) SOAP envelope.

The content of SOAP envelope is passed to a parameter called **SoapEnvelopeParam** for both requests and responses. After the function processes the SOAP envelope, make sure to store it in the same parameter.

To call the handler function, use the function name as a value for the **UserHandlerFunction** argument in the relevant Web Service Call step. For more information, see the Function Reference (**Help > Function Reference**).

Event Handler Return Codes

VuGen recognizes the following return codes for the handler function.

Return Code		Description
LR_HANDLER_SUCCEEDED	0	The Handler succeeded, but the SOAP envelope did not change.
LR_HANDLER_FAILED	1	The Handler failed and further processing should be stopped.
LR_HANDLER_SUCCEEDED_AND_MODIFIED	2	The Handler succeeded and the updated SOAP envelope is stored in SoapEnvelopeParam .

In the following example, a script handler manipulates the outgoing envelope:

```
//This function processes the SOAP envelope before sending it to the server.
int MyScriptFunction(const char* pArgs, int isRequest)
{
    if (isRequest == 1) {
        //Get the request that is going to be sent
        char* str = lr_eval_string("{SoapEnvelopeParam}");
        //Manipulate the string...
        //Assign the new request content
        lr_save_string(str, "SoapEnvelopeParam");
        return LR_HANDLER_SUCCEEDED_AND_MODIFIED;
    }
    return LR_HANDLER_SUCCEEDED;
}
Action()
{
```

```
//Instruct the web_service_call to use the handler
web_service_call( "StepName=EchoAddr_102",
    "SOAPMethod=SpecialCases.SpecialCasesSoap.EchoAddr",
    "ResponseParam=response",
    "userHandlerFunction=MyScriptFunction",
    "Service=SpecialCases",
    "Snapshot=t1174304648.inf",
    BEGIN_ARGUMENTS,
    "xml:addr="
        "<addr>"
            "<name>abcde</name>"
            "<street>abcde</street>"
            "<city>abcde</city>"
            "<state>abcde</state>"
            "<zip>abcde</zip>"
        "</addr>",
    END_ARGUMENTS,
    BEGIN_RESULT,
    END_RESULT,
    LAST);
return 0;
```

Custom Configuration Files

Configuration files let you customize advanced settings such as security information and the WSE configuration. These files let you control the behavior of the test during runtime.

The standard .NET configuration file, **mmdrv.exe.config**, is located in the VuGen installation folder. Some applications have their own configuration file, **app.config**.

You can customize the test run further, by filtering out the input or output. In addition, you can configure security information, such as token information and whether or not to allow unsigned test certificates.

For task details, see ["Customize Configuration Files" on page 771](#).

User Handler Examples

This section illustrates several common uses for user handlers.

.NET Filters

You can apply a .NET filter to your messages using the user handler mechanism.

If you are familiar with Microsoft's Web Service Enhancements (WSE) 2.0, you can create a .NET filter and register it for incoming or outgoing SOAP messages. A .NET filter is a class that is derived from `Microsoft.Web.Services2.SoapInputFilter` or `Microsoft.Web.Services2.SoapOutputFilter`. By overriding the **ProcessMessage** function of this class, you can examine and modify the envelope's body and header.

To define the filter globally for the entire script, add the following lines to the script's default.cfg file below.

```
[UserHandler]
Function=LrWsSoapFilterLoader
Args=<Filters InputFilterClass="class name" InputFilterLib="lib name"
OutputFilterClass="class name" OutputFilterLib="lib name" />
Order=BeforeSecurity/AfterSecurity/AfterAttachments
```

The **InputFilterClass** parameter indicates the name of your class, and **InputFilterLib** indicates the name of the assembly in which the class resides. For example:

```
web_service_call(
    ...
    "UserHandlerName=LrWsSoapFilterLoader",
    "UserHandlerArgs=<Filters
InputFilterClass=\"MyFilterNamespace.MyFilterClassName\"
InputFilterLib=\"MyAssemblyName\" />",
    BEGIN_ARGUMENTS,
    ...
    END_ARGUMENTS,
    ...
);
```

Use SoapOutputFilter to examine an outgoing **web_service_call** request, and SoapInputFilter to examine the response from the server. Use **InputFilterClass** and **InputFilterLib** if your filter is derived from SoapInputFilter, or **OutputFilterClass** and **OutputFilterLib** if your filter is derived from SoapOutputFilter.

To define the filter for a specific step, add the following arguments to the **web_service_call** function.

```
UserHandlerName= LrWsSoapFilterLoader
UserHandlerArgs=<Filters InputFilterClass=\"class name\" InputFilterLib=\"lib name\"
OutputFilterClass=\"class name\" OutputFilterLib=\"lib name\" />
UserHandlerOrder=BeforeSecurity/AfterSecurity/AfterAttachments
```

Overriding the Transport Layer

The following example shows a user handler function overriding the transport layer. VuGen does not automatically send the SOAP request over HTTP transport—instead it follows the transport method indicated in the custom handler.

After you receive a response, set the response envelope with the command:

```
lr_save_string(someResponseEnvelopeStr, "SoapEnvelopeParam");
```

To apply an alternate transport layer, specify **ReplaceTransport** as a value for the **UserHandlerOrder** argument. Define the transport layer in the handler.

```
web_service_call(  
    ...  
    "UserHandlerFunction=<Transport HandlerFunction>",  
    "UserHandlerArgs=<handler arguments>",  
    "UserHandlerOrder=ReplaceTransport"  
    ...  
    LAST);
```

Including MIME Attachments

When working with Web Service scripts based on the .NET toolkit, the infrastructure does not support MIME attachments. Using the handlers mechanism, you can add MIME attachment functionality to .NET scripts.

The following sections describe how to send and receive MIME attachments for the .NET toolkit. You can receive and send a MIME attachment in the same operation.

Sending MIME Attachments

To send a MIME attachment, add the boldfaced code to the **web_service_call**:

```
web_service_call( "StepName=EchoComplex_101",  
    "SOAPMethod=SimpleService|SimpleServiceSoap|EchoComplex",  
    "ResponseParam=response",  
    "Service=SimpleService",  
    "UserHandlerName=LrWsAttachmentsHandler", "UserHandlerArgs=ATTACHMENT_ADD;  
    ATTACHMENTS_FORMAT_MIME; ContentType=text/plain; FileName=C:\\temp\\results.discomap",  
    "ExpectedResponse=SoapResult",  
    "Snapshot=t1208947811.inf",  
    BEGIN_ARGUMENTS,  
    "xml:cls="  
    "<cls>"  
    "<i>123456789</i>"  
    "<s>abcde</s>"  
    "</cls>",  
    END_ARGUMENTS,  
    BEGIN_RESULT,  
    END_RESULT,  
    LAST);
```

Modify the **FileName** and **ContentType** parameters to indicate the actual path and content type.

Receiving MIME Attachments

To receive a MIME attachment, add the following code to the **web_service_call**:

```
"UserHandlerName=LrWsAttachmentsHandler",  
"UserHandlerArgs=ATTACHMENT_SAVE_ALL;ParamNamePrefix=attach;"
```

Sending and Receiving MIME Attachments

To send and receive a MIME attachment in the same **web_service_call**, modify the Web Service call as shown below:

```
"UserHandlerName=LrWsAttachmentsHandler",  
"UserHandlerArgs=ATTACHMENT_SAVE_ALL;ParamNamePrefix=attach; ATTACHMENT_ADD;  
ATTACHMENTS_FORMAT_MIME; ContentType=text/plain;  
FileName=C:\\temp\\results.discomap",
```

Prepare Scripts for Replay

This task describes how to prepare the script for replay and run it. It describes how to use the output of one Web Service call as input for another.

Assign Input Parameter Values

First save the output result to a parameter, and then reference that parameter in a later Web Service call.

1. **Save the output parameter.**
 - a. In the **Step Navigator**, double-click the Web Service call whose output you want to use, to view its properties.
 - b. In the left pane, select the output argument whose value you want to save to a parameter.
 - c. In the right pane, select **Save returned value in parameter**. Specify a name in the **Parameter** box.
2. **Use the saved parameter for input.**
 - a. In the **Step Navigator**, double-click the Web Service call whose input parameters you want to set.
 - b. In the left pane, select the input argument for which to use the saved parameter.
 - c. In the right pane, select **Value**, and click on the abc icon. The Select or Create Parameter box opens.
 - d. Select the saved output parameter from the **Parameter name** list.
 - e. To specify an input parameter in Script view, select the value you want to replace and select **Use Existing Parameters** from the shortcut menu. Select one of the available parameters.

Set the Runtime Settings - Optional

Open the runtime settings (F4) to configure JMS and VM settings. Click the **JMS > Advanced** node. For details, see the **JMS > Advanced** view in the runtime settings.

Configure XSDs With any type Elements - Optional

For Web Services that have an XSD schema with an **Any** type element, `<xsd:element name="<Any_element>" type="xsd:anyType" />`, check that the script conforms with the following model:

```
BEGIN_ARGUMENTS,  
    "xml:Any_element="  
    "<Any_element>"  
    "<string>the string to send</string>"  
    "</Any_element>",  
END_ARGUMENTS,
```

The actual SOAP may differ slightly, but as long as your script conforms to the above model, it will run properly.

You can also send complex type elements for the **<any>** type. For example:

```
"xml:Any_element="  
    "<Any_element>"  
    "<myComplexTypeName>"  
    "<property1>123</property1>"  
    "<property2>456</property2>"  
    "</myComplexTypeName>"  
    "</Any_element>",
```

Run the Script

Click **Replay > Run**. Observe the output log for relevant messages.

Review the Test Results

Display the Replay Summary to review results of the test run. For details, see ["Replay Summary Pane" on page 108](#).

Send Messages over JMS

This task describes how to send messages using the JMS transport method.

1. Open the step properties

In the **Step Navigator**, select the step whose transport you want to set, and then select **Show Arguments** from the shortcut menu.

2. Select the JMS transport method

Select the **Transport Layer Configuration** node and choose **JMS Transport**.

For UI details, see ["New Web Service Call Dialog Box" on page 740](#).

3. Set the runtime settings - optional

Configure the runtime settings as described in the **JMS > Advanced** view.

4. Send synchronous JMS messages - optional

Once you create a Web Service call and designate the transport method as JMS, VuGen sends the

JMS messages in a synchronous manner. If desired, specify the queue information.

5. **Send asynchronous JMS messages - optional**

To implement asynchronous messages over JMS, you send the request or retrieve the response using JMS steps—not Web Service calls.

- a. Click within the script at the desired location. Select **Insert > New Step** from the right-click menu, and locate the **JMS Functions** in the Steps Toolbox.
- b. Select a JMS function: **JMS Send Message Queue** sends a message to a queue. **JMS Receive Message Queue** receives a message from the queue.
- c. Click **OK** to open the JMS function properties.
- d. Specify a queue name and click **OK** to generate the JMS functions.

For additional information about these functions, see the Function Reference (**Help > Function Reference**) (or click **F1** on the function).

6. **Send messages over JMS using SOAP messages - optional**

To send messages over JMS, using the SOAP message and without a Web Service call:

- a. Record SOAP messages using a standard Web protocol.
- b. Click within the script at the desired location. Select **Insert > New Step** from the right-click menu, and locate the **JMS Functions** in the Steps Toolbox.
- c. Select a JMS function: **Send Message Queue** or **JMS Receive Message Queue**.
- d. Click **OK** to open the JMS function properties.
- e. Specify a queue name and click **OK** to generate the JMS functions.

For details, see the Function Reference (**Help > Function Reference**) (or click **F1** on the function).

Send Messages over HTTP/S

This task describes how to send messages using the HTTP transport method.

1. Open the step properties.

In the **Step Navigator**, select the step whose transport you want to set, and then select **Show Arguments** from the shortcut menu.

2. Select the HTTP/S transport method.

Select the **Transport Layer Configuration** node and choose **HTTP/S Transport**.

3. Send a HTTP synchronous message. (Optional)

To send messages in synchronous mode over HTTP, create a standard Web Service call, and do not enable the **Async Support** option.

4. Send asynchronous HTTP messages. (Optional)

- a. Choose **HTTP/S Transport** and select the **Async Support** option.
- b. Type an event name in the **Async Event** box.

- c. Click **OK** to generate the Web Service call.
- d. Add a **Wait for Event** step. Select **Insert > New Step** from the right-click menu and choose **web_service_wait_for_event** from the SOAP functions in the Steps Toolbox.
- e. Specify a step name, a quantifier, and a timeout. Click **Add** and insert the name of the event that you defined in the previous step.

In Script view, VuGen indicates asynchronous messaging with the added parameter, **AsyncEvent**.

```
web_service_call("StepName=EchoString_101",
    "SOAPMethod=EchoRpcEncoded.EchoSoap.EchoString",
    "ResponseParam=response1",
    "Service=ExtendedECHO_rpc_encoded",
    "AsyncEvent=Event_1",
    "Snapshot=t1157371707.inf",
    BEGIN_ARGUMENTS,
    "sec=7",
    "strString=mytext",
    END_ARGUMENTS,
    BEGIN_RESULT,
    "EchoStringResult=first_call",
    END_RESULT,
    LAST);
```

The **AsyncEvent** flag instructs the Vuser to wait for the response of previous asynchronous service requests.

5. Send an asynchronous message using WS-Addressing. (Optional)
 - a. Select the **Async Support** option and provide an event name in the **Async Event** box. This can be an arbitrary name.
 - b. Select **WSA Support**. In the **WS-A Reply to** box, enter an IP address or **autodetect** to use the current host. Autodetect is useful when running the same script on several different machines. The server will reply to the specified location when the event occurs.
 - c. Click **OK** to save the settings.
 - d. Instruct the Vuser to wait for an event. Select **Insert > New Step** from the right-click menu and choose **web_service_wait_for_event** from the SOAP functions in the Steps Toolbox.
 - e. Specify a step name, quantifier, and timeout. To add an event name, click **Add**. The Web Service will wait for the specified event before responding.
 - f. Use the **Edit**, **Move Up**, and **Move Down** buttons to manipulate the events.

Define a Testing Method

This task describes how to select a testing method.

1. Open the step properties

In the Step Navigator, right-click the step whose response you want to test, and select **Show Arguments**.

2. Select an argument

Select the Output Argument node. For details, see ["New Web Service Call Dialog Box" on page 740](#).

3. Select a testing method and choose an expected response

- To perform negative testing only, select the **Negative Testing** check box and choose **SOAP Fault** as the **Expected Response**.
- To accept any type of SOAP response, select the **Negative Testing** check box and choose **Any SOAP** as the **Expected Response**.
- To perform positive testing only, clear the **Negative Testing** check box.

4. Verify function in the script

In Script view, VuGen indicates the testing method with the **ExpectedResponse** argument. In the following example, the script performs negative testing, indicated by the **SoapFault** value:

```
web_service_call("StepName=AddAddr_101",
    "SOAPMethod=AddrBook | AddrBookSoapPort | AddAddr",
    "ResponseParam=response",
    "Service=AddrBook",
    "ExpectedResponse=SoapFault",
    "Snapshot=t1189409011.inf",
    BEGIN_ARGUMENTS,
    END_ARGUMENTS,
    BEGIN_RESULT,
    END_RESULT,
    LAST);
```

5. Evaluate the SOAP fault value

When you replay a script that results in a SOAP fault, VuGen saves the fault to a parameter called **response**. To check the returned value of the SOAP fault, evaluate the **response** output parameter using **lr_xml_find**.

In the following example, **lr_xml_find** checks for a **VersionMismatch** SOAP fault and issues an output message.

```
lr_xml_find("XML={response}",
    "FastQuery=/Envelope/Body/Fault/faultString ",
    "Value=VersionMismatch",
    LAST);
if (soap_fault_cnt > 0)
    lr_output_message("A Version Mismatch SOAP Fault occurred")
```

For more information about **lr_xml_find**, see the Function Reference (**Help > Function Reference**).

Add a Database Connection

This task describes how to add a database connection step through Tree view.

1. Open Solution Explorer

Select **View > Solution Explorer**.

2. Select a section

Select the desired section: **vuser_init** or **Action**. To avoid repeating the connection sequence in every iteration, place it in the **vuser_init** section.

3. Insert a database connection step

Select **Design > Insert in Script > New Step**. Choose the **lr_db_connect** step. The Database Connection dialog box opens. Specify a **Step Name**, **Connection Name**, and **Data Provider**, OLEDB or SQL.

4. Create a database connection string

- a. Click **Connection String Generator** to generate a database connection string specific to your environment.
- b. Indicate the connection properties:
 - **Server Name**
 - **Database Name**
 - **Authentication** method: Windows Authentication or User/password.
 - **Username** and **Password**
- c. Click **Test Connection** to verify that the information you provided is correct.
- d. Select an **SQL Provider**, OLEDB or SQL, and click **Generate**.

5. Verify function in the script

Check that an **lr_db_connect** function was written to the script.

Create a User Handler

This task describes how to write a user handler for your script.

1. Prerequisite: Create a Web Service call

Import a WSDL file and create a standard Web Service Call. For details, see ["Adding Web Service Script Content - Overview" on page 725](#).

2. Define a user handler function

Define a user handler before the Web Service call:

```
int MyScriptFunction(const char* pArgs, int isRequest)
```

```
{  
...  
}
```

3. Call the user handler function

Call the handler function by specifying the function name as a value for the **UserHandlerFunction** argument. in the Web Service Call.

```
web_service_call(  
...  
"UserHandlerFunction=MyScriptFunction",  
"UserHandlerArgs=<handler arguments>",  
LAST);
```

4. Evaluate the handler function

Evaluate the handler's return code to determine if it succeeded. Use the return codes as described in ["User Handlers" on page 758](#).

```
//This function processes the SOAP envelope before sending it to the server.  
int MyScriptFunction(const char* pArgs, int isRequest)  
{  
    if (isRequest == 1) {  
        //Get the request that is going to be sent  
        char* str = lr_eval_string("{SoapEnvelopeParam}");  
        //Manipulate the string...  
        //Assign the new request content  
        lr_save_string(str, "SoapEnvelopeParam");  
        return LR_HANDLER_SUCCEEDED_AND_MODIFIED;  
    }  
    return LR_HANDLER_SUCCEEDED;  
}
```

5. Create a DLL file. (Optional)

To define a user handler through a DLL, locate the API header file, **LrWsHandlerAPI.h** in the product's **include** folder.

You can use a sample Visual Studio project located in the samples/WebServices/SampleWsHandler folder as a template for creating a handler. The sample retrieves the request and response envelope and saves it to a parameter. To use this sample, open it in Visual Studio and modify it as required. If you do not need to save the request/response to a parameter, you can remove that section of the sample.

After editing the sample, save it and compile the DLL. When you compile the project, Visual Studio places the **<user_handler_name>.DLL** file in the **bin** folder. If you compile the project from another location, or if you want to copy the DLL from one machine to another, make sure to place it in the **bin** folder.

6. Configure the user handler. (Optional)

Declare the DLL user handler globally or locally.

To apply the user handler globally to all requests in the script, add the following section to the **default.cfg** file in the script's folder.

```
[UserHandler]
Function=<name>
Args=<arguments>
Order=<BeforeSecurity/AfterSecurity/AfterAttachments>
```

- **Name.** The name of the DLL.
- **Args.** A list of the configuration arguments for the handler. Use the **GetArguments** method to retrieve the arguments in your handler.
- **Order.** The order in which Vusers process the user handler in requests: **Before Security**, **After Security**, or **After Attachments**. You can also use this argument to override the transport layer, by entering the value **Replace Transport**.

Note: Setting the **UserHandlerFunction** property of a **web_service_call** function, overrides the definitions in the .cfg file.

By default, user handlers are processed before the security. For request messages, Vusers process the attachments handler after the security handler. For responses, Vusers process the handlers in a reversed order. In typical cases the order does not matter, so any value is acceptable.

To override the Transport layer, specify **Order=Replace Transport** and specify the new transport handler. If you implement the transport handler as a separate DLL, the **HandleRequest** function is called, while the **HandleResponse** function is ignored.

To use the handler locally, for a specific request, add the following arguments to the **web_service_call** function:

```
UserHandlerName=<name1>
UserHandlerArgs=<args1>
UserHandlerOrder=<BeforeSecurity/AfterSecurity/AfterAttachments/Replace
Transport>
```

7. Copy the user handler to all required machines

Make sure that the user handler DLL is accessible to all Load Generator machines running scripts that call it. You may, for example, copy it to the product's **/bin** folder.

If you copy the script to another machine, it retains the handler information, since it is defined in script's folder.

8. Implement the user handler. (Optional)

To implement a user handler, you use the entry functions **HandleRequest** or **HandleResponse**.

Both functions have a single parameter, **context**, whose properties you can set in your handler. Use the Get functions to retrieve properties, and Set functions to pass information from the replay framework to the handlers or between the handlers.

- **GetEnvelope.** Gets the envelope content. For example:

```
const char * pEnvelope = context->GetEnvelope();
```
- **GetEnvelopeLength.** Gets the envelope length
- **SetEnvelope.** Sets the envelope content and length. For example:

```
string str("MySoapEnvelope...");  
context->SetEnvelope(str.c_str(), str.length());
```
- **SetContentType.** Sets a new value for HTTP header content type
- **LogMessage.** Issues a message to the replay log
- **GetArguments.** Gets the configuration arguments defined for the current handler in order to pass it to the DLL
- **GetProperty.** Gets a custom property value
- **SetProperty.** Sets a custom property value

For more information, see the comments in the **LrWsHandlerAPI.h** file located in the product's **include** folder.

Customize Configuration Files

The following steps describe how to modify configuration files. For details, see "[Custom Configuration Files](#)" on page 760.

Locate the configuration file

Determine the location of the configuration file. The standard .NET configuration file, **mmdrv.exe.config** is located in the product's **bin** folder. Some applications have their own file, **app.config**.

Save the application's configuration file

If your application has its own app.config file:

- To apply the configuration information globally to all scripts, save the **app.config** file as **mmdrv.exe.config** in the **bin** folder, overwriting the existing file.
- To apply the configuration information locally, specifically for this script, copy the **app.config** file to the script's folder. This overrides the **mmdrv.exe.config** file, and remains associated with this script even when you copy it to other machines.

Set the security - optional

By default, VuGen allows unsigned certificates to facilitate testing. To disallow unsigned certificates, modify the **allowTestRoot** flag in the <security> section to **false**.




```
<security>  
  <x509 storeLocation="currentuser" alllowTestRoot="false"
```

Web Services Snapshots - Overview


User scripts based on the Web Services protocol utilize VuGen's Snapshot pane.

The Snapshot pane enables you to view snapshots of Web service calls. When you display the Snapshot pane for a Web Services script, the left side of the Snapshot pane displays a tree view of the snapshot data; the right side of the Snapshot pane displays a text view of the snapshot data.







The tree view on the left of the Snapshot pane is composed of a number of nodes. An icon to the left of each node indicates the type of the node:

-  **Element:** Indicates that the node represents an element in the XML file.
-  **Attribute:** Indicates that the node represents an attribute in the XML file.
-  **Value:** Indicates that the node represents a value in the XML file.

In addition to the basic Snapshot pane functionality, the Snapshot pane for Web Services scripts includes additional functionality. The UI for this additional functionality is described below.

To access	Select View > Snapshot , or click the Show Snapshot Pane button  on the VuGen toolbar.
Relevant tasks	"Prepare Scripts for Replay" on page 763

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
 Response	Displays a snapshot of the SOAP response returned by the server.
 Request	Displays a snapshot of the SOAP request sent to the server by the Web Service call.
	Opens the XPath Search dialog box which enables you to perform an XPath search of the snapshot.
	Displays the previous or next results of the XPath search.
	Displays or hides the XML attribute nodes in the tree view of the snapshot.
	Displays or hides the XML value nodes in the tree view of the snapshot.

**<shortcut
menu>**

- **Copy Selection.** Copies the text that is selected in the text view to the clipboard.
- **Search Community.** Performs a community search using the text that is selected in the text view as the search string. For details about performing a community search, see ["Editor Pane" on page 54](#).
- **Copy XPath.** In the tree view, copies the XPath of the selected node to the clipboard. In the text view, copies the XPath of the XML element in which the cursor is located to the clipboard.
- **Copy full value.** In the tree view, copies the full XML code of the selected node to the clipboard. In the text view, copies the full XML code of the XML element in which the cursor is located.
- **Insert XML Check.** Opens the Insert XML Check dialog box that enables you to insert an **XML Find** step into the Vuser script.





Note:

- This option is available in the Response view only.
- This option is available for **attribute**  and **value**  nodes only.

- **Save value in parameter.** Opens the Save Value as Parameter dialog box that enables you to save the selected value to a simple parameter.





Note:

- This option is available in the Response view only.
- This option is available for **attribute**  and **value**  nodes only.



- **Save XML in parameter.** Opens the Save Value as Parameter dialog box that enables you to save the selected value to an XML parameter.
This option is available in the Response view only.
- **Create Correlation.** Opens the Correlation tab in the Design Studio. The text selected in the Snapshot pane appears as a manual correlation entry in the Design Studio. For details, see ["Manually Correlate Scripts" on page 242](#).



Note:

- This option is available in the Response view only.
- This option is available for **attribute**  and **value**  nodes in the tree view, and when text is selected in the text view.

- **Create Correlation Rule.** Opens the Add as Rule dialog box that enables you to add the selected text as part of a correlation rule. For details, see ["Correlation Tab \[Design Studio\] Overview" on page 234](#).

	<p>Note:</p> <ul style="list-style-type: none"> • This option is available in the Response view only. • This option is available for attribute  and value  nodes in the tree view, and when text is selected in the text view.
--	--

See also:

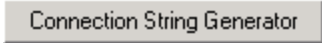
- ["Work with Snapshots" on page 279](#)
- ["Snapshot Pane" on page 62](#) for details on the standard Snapshot pane UI

Database Connection Dialog Box

This dialog box helps you create a string to connect to your database.

To access	Click Connection String Generator in the Database Connection dialog box.
Relevant tasks	"Send Messages over JMS" on page 764
See also	"Connection String Generator Dialog Box" below

User interface elements are described below:


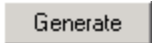
UI Element	Description
	Opens the Connection String Generator. For details, see "Connection String Generator Dialog Box" below .
Step Name	The name or IP address of the database server.
Connection String	The string by which to connect to the database. Use the Connection String Generator .
Data Provider	The SQL provider: OleDb or SQL .

Connection String Generator Dialog Box

This dialog box helps you create a string to connect to your database.

To access	Click Connection String Generator in the Database Connection dialog box.
Relevant tasks	"Send Messages over JMS" on page 764
See also	"Database Connection Dialog Box" above

User interface elements are described below:

UI Element	Description
	Tests the connection to the database.
	Generates the database connection string and writes it in the Connection String field in the Database Connection dialog box.
Server Name	The name or IP address of the database server.
DB Name	The name of the database.
Authentication	The authentication method for the database: Windows Authentication or User/password . <ul style="list-style-type: none">• User Name, Password. The credentials for the database.
SQL Provider	The SQL provider: OLEDB or SQL .

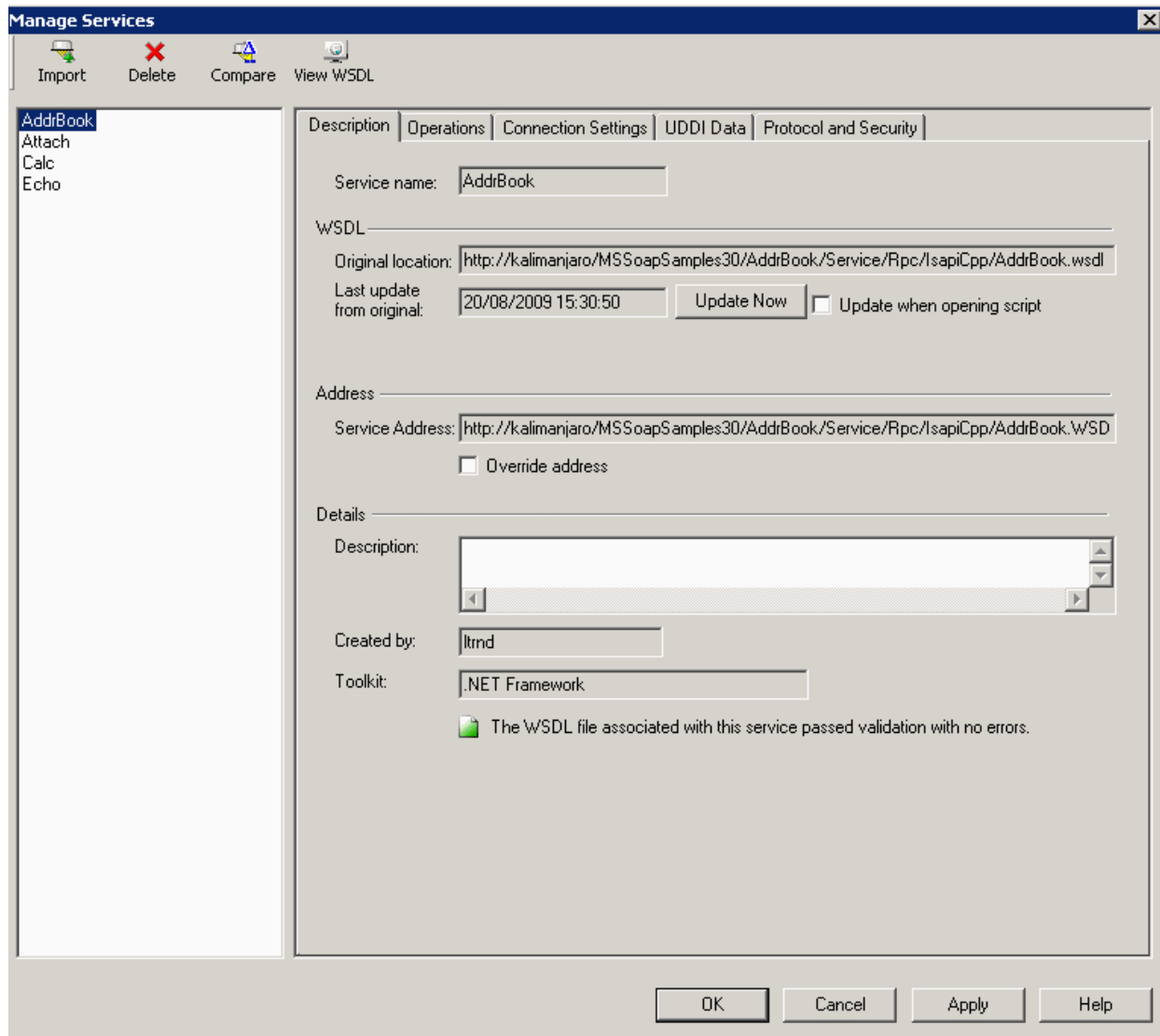
Web Services - Managing Services

Managing Services Overview

The Service Management window lets you manage a list of service entries for the current script. You can view and set the properties of each service entry.

You add service entries to the list by importing WSDL files. When you add a WSDL to the list, VuGen creates a working copy that it saves with the script—it is not global. Therefore, for each script that you create, you must import the desired WSDL files.

The Service Management window provides quick access buttons for importing, deleting, and comparing services.



Description tab

The **Description** tab displays information about the service:

- **Original location.** The original source of the WSDL file (read-only).
- **Service name.** The name of the Web Service (read-only).
- **Last update from original.** The last date that the local copy was updated from the original source (read-only). You can update the version manually or retrieve it automatically each time you reopen the test.
- **Service address.** An endpoint address to which the request is sent. If required, you can override the endpoint specified in the WSDL file.
- **Created by.** The name of the user who originally imported the service (read-only).
- **Toolkit.** The toolkit associated with the script. You set this before importing the first WSDL file (read-only).

[Manage Services" on page 779.](#)

This section also includes:

- ["Importing Services" below](#)
- ["Comparison Reports" below](#)

Importing Services

VuGen lets you import services for the purpose of creating a high-level tests with Web Service Call steps. Typically, you begin creating a script by importing a WSDL file.

The Import mechanism requires the following information:

- **Source.** The source of the WSDL: URL, File, UDDI, or Application Lifecycle Management. UDDI is a universal repository for services (Universal Description, Discovery, and Integration). Service brokers register and categorize published Web Services and provide search capabilities. The UDDI business registry is an example of a service broker for WSDL-described Web Services.
- **Location.** the path or URL of the WSDL, entered manually or by browsing.
- **Toolkit.** The toolkit to permanently associate with all services in the script for all subsequent imports and replays (only available for the first service added to the script). The toolkit setting instructs VuGen to send real client traffic using an actual toolkit—not an emulation.

VuGen supports the .NET Framework with WSE 2 version SP3 and Axis/Java based Web Services Framework toolkits. VuGen imports, records, and replays the script using the actual .NET or Axis toolkit. By default, VuGen uses automatic detection to determine the most appropriate toolkit.

- **Connection Settings.** Authentication or proxy server information. This setting is useful for WSDLs residing on secure servers, or WSDLs that must be accessed via a proxy server.

If VuGen detects a problem with your WSDL when attempting to do an import, it issues an alert and prompts you to open the report. The report lists the errors and provides details about them.

For task details, see ["Add and Manage Services" on the next page.](#)

Comparison Reports

VuGen lists the differences between the files in a Comparison report.

You can configure the comparison settings, indicating which items to ignore during the comparison. For more information, see the ["XML/WSDL Comparison Dialog Box" on page 786.](#)

In WSDL Comparison reports, there are two columns— **Working Copy** and **Original File**. The Working Copy is the WSDL stored with the script, while the Original File is the WSDL at its original location—a network file path or a URL.

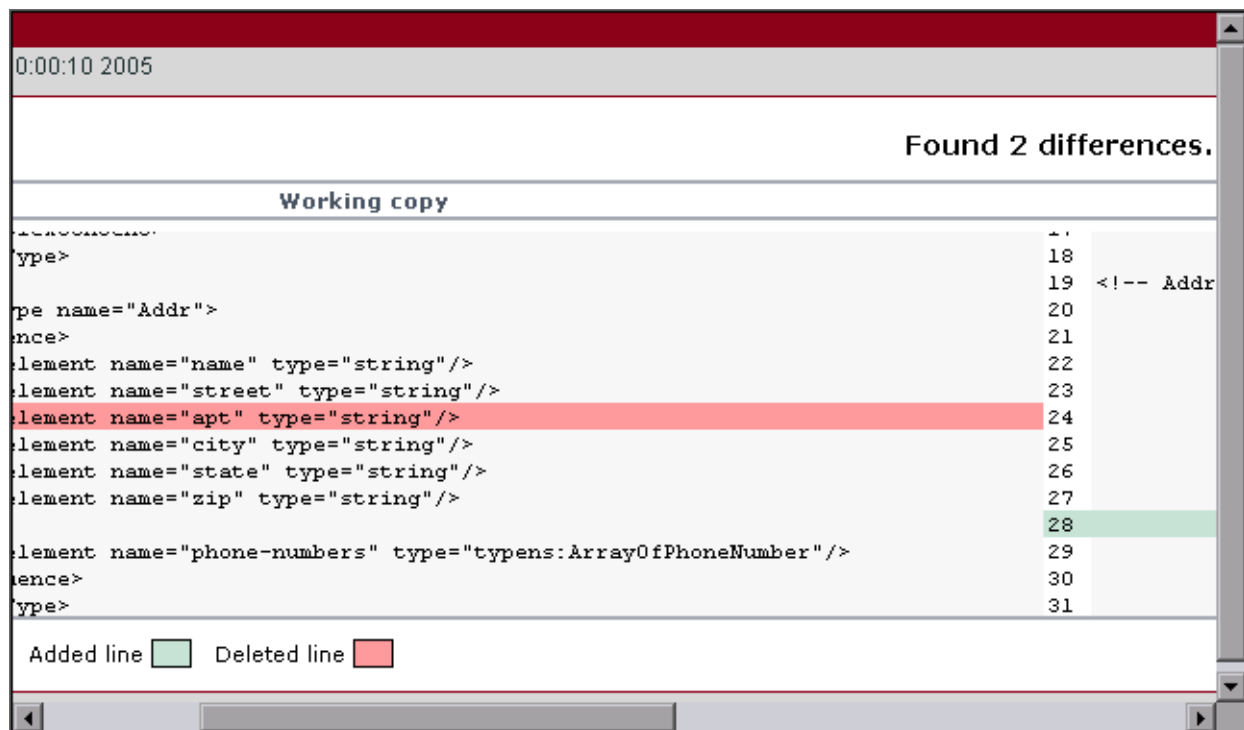
In XML Comparison reports, each column displays the path of an XML file.

The Comparison report uses the following legend to mark the differences between the two files:

- **Yellow.** Changes to an existing element (shown in both versions).

- **Green.** A new element added (shown in the original file copy).
- **Pink.** A deleted element (shown in the working copy).

In the following example, line 24 was deleted from the original copy and line 28 was added.



Web Reference Analyzer

Many WSDL files reference other files such as XSD and other XML files. Before running a script, you may want to determine what these files are and if they are available.

VuGen's WSDL Reference Analyzer checks the WSDL for dependencies, and lists them in the WSDL Reference Analyzer window and in a log file.

The Analyzer places the WSDL and its dependent files in a zip archive file. It saves the dependency information to a log file, listing its path in the Analyzer window.

For user interface details, see ["WSDL Reference Analyzer Dialog Box" on page 786](#).

For task details, see ["Analyze WSDL Dependencies" on page 781](#).

Add and Manage Services

This task describes how to create a list of services that you can call from your test. Using the Manage Services window, you import services and configure their settings.

Open the Manage Services Dialog Box

Select **SOA Tools > Manage Services** or click the toolbar button to open the Manage Services dialog box.

Import a Service

Click **Import**. In the Import Service dialog box, select a WSDL source and browse to the location.

For **URL** type imports, the Browse button opens a new browser. Navigate to the WSDL and then close the browser. This action places the URL in the location box. For details, see the ["Import Service Dialog Box" on page 784](#).

If your service requires authentication or uses a proxy, configure these settings before importing the WSDL. Expand the Import Services dialog box and click **Configure**. For details, see the ["Connection Settings Dialog Box" on page 784](#).

Repeat this step for all the services you want to include in your test.

Get to Know the WSDL

Familiarize yourself with the WSDL. View its details as described in the ["Manage Services Dialog Box" on the next page](#).

Click **View WSDL** to open the locally saved WSDL file in Internet Explorer and study its structure.

Check for WSDL Updates - Optional

Use the Comparison tool to check that the WSDL did not change since your last import or update.

First, set the comparison options. Click **SOA Tools > SOA Settings > XML/WSDL Comparison**. Specify what differences to ignore. For details, see ["XML/WSDL Comparison Dialog Box" on page 786](#).

In the Manage Services window, click **Compare** to open a report comparing the working copy of the WSDL with the one at the original location.

If you discover changes in the Comparison report, click **Update Now** to retrieve the latest version of the WSDL from its source.

Override the Service Address- Optional

View the address in the **Service Address** box. This is the default endpoint address as retrieved from the WSDL. If you want to override it, select **Override address** and type in an alternate endpoint address for the service requests.

To return to the default address, clear the **Override address** option. For details, see the ["Manage Services Dialog Box" on the next page](#).

Set a Security Scenario - Optional

Click the **Protocol and Security** tab to use WS-Security or another type of a security scenario.

For more information, see ["Web Services - Security" on page 787](#).

Analyze WSDL Dependencies

This task describes how to use the Reference Analyzer to determine WSDL dependencies. For user interface details, see ["WSDL Reference Analyzer Dialog Box" on page 786](#).

1. Open the Reference Analyzer

Select **SOA Tools > WSDL Reference Analyzer**.

2. Select a source and target

In the **Select WSDL file** box, indicate the location of the WSDL you want to analyze.

In the **Output file path** box, indicate a location for the zip file.

3. Begin the analysis


Click **Start Analyzing**. The Analyzer lists all of the dependencies in the output window along with their paths.

4. View the log



View the results in the log window. To clear the results and perform another analysis, click **Clear Log**.



Manage Services Dialog Box

This dialog box enables you to manage your WSDLs, provide authentication information, and set a security scenario.

To access	<p>Use one of the following:</p> <ul style="list-style-type: none"> Click the Manage Services button  on the VuGen toolbar. SOA Tools > Manage Services
Relevant tasks	"Add and Manage Services" on page 779


User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Opens the Import Service dialog box.
	Removes the selected service from the list.

	Opens the WSDL Comparison Report showing the Working copy and Original copy of the WSDL side-by-side. To set the comparison settings, see "XML/WSDL Comparison Dialog Box" on page 786 .
	Displays the WSDL in a browser.
<WSDL list>	A list of the imported WSDLs.
Description tab	Provides information about the WSDL, its endpoint address, toolkit, and update information.
Operations tab	Lists the operations of the service: Operation Name , PortName , and Used in Script (Yes or No). Click a column to sort the operations by that column's data. Click it again to reverse the sorting order.
Connection Settings tab	Allows you to provide authentication settings for the machine from which you are importing a service. Note: This only applies to URL and UDDI type imports.
UDDI Data tab	The UDDI server, UDDI version, and service key.
Protocol and Security tab	Allows you to view and set a security scenario for your Web Service calls. For more information, see below.

Description Tab

The following elements are displayed in the **Description** tab:

UI Element	Description
	Loads the latest version of the WSDL from its original location.
Created By	The name with which you logged in. You can edit this field and specify a different name. This is useful for sorting the services in reports (read-only).
Description	An editable field into which you can type information about the service.
Last update from original	The last date and time the WSDL was updated (read-only).
Original Location	The original location from where the WSDL was imported (read-only).

Override address	Enables you to enter an alternate endpoint for the service in the Service Address box.
Service Address	The endpoint of the service to which service requests are sent, retrieved from the WSDL file (read only). To override the default address, select Override address .
Service Name	The native service name in the WSDL file that is displayed by default when importing the service (read-only).
Toolkit	The toolkit associated with the service. You set this when you import the service (read-only).
Update when opening script	Updates the WSDL from its source each time you open the script.

Connection Settings Tab

The following elements are included:

UI Element	Description
Authentication	<p>Use Authentication Setting: Enables you to enter credentials for authentication.</p> <ul style="list-style-type: none"> • Username, Password. The user name and password to use for retrieving the WSDL. <p>Tip: For users not in the default domain, type the domain name before the user name. For example, domain1/alex_qc.</p>
Proxy	<p>Use Proxy Setting. Enables you to enter proxy details and credentials.</p> <ul style="list-style-type: none"> • Server. Name or IP address of proxy server. • Port. Port through which to access the WSDL. • Username, Password. the user name and password to be used for authentication. For users not in the default domain, type the domain name before the user name. For example, domain1/alex_qc.

UDDI Data Tab

The following elements are included:

UI Element	Description
Service Key	A unique identifier of the service on the UDDI server, used to locate the service definition when updating the service.

UDDI Server	The URL address and version of the UDDI server from which the service definition is imported.
UDDI Version	The version of the UDDI registry: 2 or 3.

Connection Settings Dialog Box

Enables you to provide authentication credentials and proxy server details for the machine hosting the WSDL file.

To access	<ul style="list-style-type: none"> For a new service: Select SOA Tools > Manage Services. Click the Import button. In the Import Services dialog box, click Connection Settings. For existing services: Select a service in the Mange Services dialog box, and click the Connection Settings tab.
Important information	Only available for services imported through a URL and UDDI.
Relevant tasks	"Add and Manage Services" on page 779

The following elements are included:


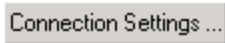
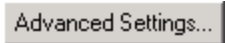
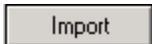
UI Element	Description
Authentication	<p>Use Authentication Setting: Enables you to enter credentials for authentication.</p> <ul style="list-style-type: none"> Username, Password. the user name and password to use for retrieving the WSDL. <p>Tip: For users not in the default domain, type the domain name before the user name. For example, domain1/alex_qc.</p>
Proxy	<p>Use Proxy Setting. Enables you to enter proxy details and credentials.</p> <ul style="list-style-type: none"> Server. Name or IP address of proxy server. Port. Port through which to access the WSDL. Username, Password. the user name and password to be used for authentication. For users not in the default domain, type the domain name before the user name. For example, domain1/alex_qc.

Import Service Dialog Box

Enables you to import WSDLs from a file system, a URL, Application Lifecycle Management, a UDDI, or Systinet.


To access	Use one of the following: <ul style="list-style-type: none"> • Select Services > New > Import Services • Select New > Import Services from the shortcut menu
Relevant tasks	"Add and Manage Services" on page 779

The following elements are included:

UI Element	Description
	Browse. Enables you to locate a service on the file system, through a browser, UDDI registry, or Application Lifecycle Management repository depending on your Import WSDL from selection.
	Opens the Connections Settings dialog box for configuring the authentication and proxy settings of the server hosting the WSDL. For details, see "Connection Settings Dialog Box" on the previous page .
	Allows you to select a toolkit for the test. Choose Automatic , .NET , or Axis . The Automatic setting uses an algorithm to determine the most suitable toolkit.
	Begins the import process.
Select WSDL from	Location of WSDL. Browse for the information or enter it manually: <ul style="list-style-type: none"> • URL: Complete URL. Make sure to insert a complete URL—not a shortened version. • File: Full path and file name. • UDDI: UDDI registry ID. The Browse button opens the "Search for Service in UDDI Dialog Box" below.

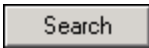
Search for Service in UDDI Dialog Box

This dialog box enables you to locate a specific service from a UDDI registry.

To access	<ul style="list-style-type: none"> • In the Manage Services window, click Import. • In the Import dialog box, select UDDI in the Select WSDL from section. • Click .
Relevant tasks	"Add and Manage Services" on page 779

User interface elements are described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
------------	-------------

	Begins the search for a service based on the text in the All or part of the service name box.
<service list>	An alphabetical list of all the services that match the string. The grid shows the following columns: Service Name , Service Key , Service Description , Service WSDL .
All or part of the service name	A string including the desired service name or part of the name. You do not need to use wildcard expressions. The following options narrow the search: <ul style="list-style-type: none"> • Exact Match. The service name must exactly match your text. • Case Sensitive. The case of service name must match the case of the specified text.
UDDI server inquiry address	The complete path for the inquiry on the UDDI server.
UDDI Version 2/3	The UDDI version of the services to display in the list.

XML/WSDL Comparison Dialog Box

This dialog box enables you to configure the settings for comparing different versions of a WSDL. You can instruct the comparison tool to ignore specific differences such as case, comments, and so forth.

To access	SOA Tools > SOA Settings > XML/WSDL Comparison.
Relevant tasks	"Add and Manage Services" on page 779

User interface elements are described below:

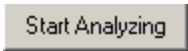
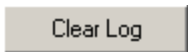

UI Element	Description
Show only differences	Show only differences in the report—do not display the matching text.
Ignore case	Do not show case mismatches as differences.
Ignore comments	Do not mark mismatches in the comment as differences.
Ignore processing instructions	Do not mark mismatches in the processing instructions as differences.
Ignore namespaces	Do not mark mismatches in namespaces as differences.

WSDL Reference Analyzer Dialog Box

This dialog box enables you to determine the dependencies of a WSDL file.

To access	SOA Tools > WSDL Reference Analyzer
Relevant tasks	"Analyze WSDL Dependencies" on page 781

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
	Begins the analysis, showing all the results in the Log window.
	Clears the log window and log file.
	Opens the folder containing the output file.
<log window>	A running log of the reference analysis.
Select WSDL file	The local path or URL of the WSDL file to analyze.
Output file path	A location for the output zip file.

Web Services - Security

Setting Security Overview

When building Web Service applications, there is a challenge in building scalable applications that are secure. You can secure Web Services by having the message sent over a secure transport, such as Secure Sockets Layer (SSL), but this is limited to point-to-point communication.

To allow you to send your messages securely, VuGen supports several security mechanisms, Security Tokens (WS-Security), and SAML.

The following table lists the considerations for using each of the models.

Legacy Model	Scenario Based Model
You are working with a script that already uses the legacy model	You are testing a WCF Service.
You are testing a service written in frameworks such as .NET 2.0, Axis, or other older toolkits	You are testing a service written in a new framework, such as Axis2 or Metro (WSIT).
You require a low-level control over WS-Security tokens	Your service uses advanced specifications such as WS-SecureConversation or WS-Trust.
You are having trouble using the new model or find the capabilities of the legacy more adequate for your needs	You are having trouble using the legacy model or you find the capabilities of the new model more adequate.

Note: If your WSDL is located in a secure location, you must provide the security information through the Manage Services dialog box. For more information, see the ["Connection Settings Dialog Box" on page 784](#).

See also:

- ["Security Tokens and Encryption" below](#)
- ["SAML Security Options" on page 790](#)

Security Tokens and Encryption

The WS-Security specification lets you place security credentials in the actual SOAP message. You accomplish this by instructing a client to obtain security credentials from a source that is trusted by both the sender and receiver. When a SOAP message sender sends a request, those security credentials, known as security **tokens**, are placed in the SOAP message. When the Web server receives the SOAP request, it does not need to send additional requests to verify the integrity of the sender. The server verifies that the credentials are authentic before letting the Web Service execute the application. By not having to go back to the source of the credentials, this significantly improves the application's scalability.

To further secure Web Services, it is common to use digital signatures or encryption for the SOAP messages. Digitally signing a SOAP message verifies that the message has not been altered during transmission. Encrypting a SOAP message helps secure a Web Service by making it difficult for anyone other than the intended recipient to read the contents of the message.

The Web Services security mechanism associates security tokens with messages. This mechanism supports several security token formats to accommodate a variety of authentication requirements. For example, a client might need to provide a proof of identity or a security certificate.

To support WS-Security, VuGen allows you to create security tokens for your script. You can create multiple tokens and set their properties. After creating a token, you use it to sign or encrypt a SOAP message.

In certain instances, you do not send the token explicitly—you use the token for the purpose of signatures or encryption, without including the actual token in the SOAP envelope header. Using the **Security Token** option, you can indicate whether to send the actual token explicitly.

Available Security Tokens

The available tokens are **Username and Password**, **X.509 Certificate**, **Kerberos**, **Kerberos2**, and **PFX File**. The information you need to provide differs for each token.

- **UserName Token.** The **UserName** token contains user identification information for the purpose of authentication: **User Name** and **Password**.
You can also specify Password Options, indicating how to send the password to the server for authentication: **SendPlainText**, **SendNone**, or **SendHashed**.
- **X.509 Certificate Token.** This security token is a token based on an X.509 certificate. To obtain a

certificate, you can either purchase it from a certificate authority, such as VeriSign, Inc. or set up your own certificate service to issue a certificate. Most Windows servers support the public key infrastructure (PKI) which enable you to create certificates. You can then have it signed by a certificate authority or use an unsigned certificate.

When you add an X.509 token to the Vuser script, you specify the **Token Name**, **Certificate**, and **Reference type**. The Browse button opens the ["Select Certificate Dialog Box" on page 815](#) which allows you to find a certificate from a Windows store.

- **Kerberos / Kerberos2 Tokens.** (for Windows 2003 or XP SP1 and later) The Kerberos protocol is used to mutually authenticate users and services on an open and unsecured network. Using shared secret keys, it encrypts and signs user credentials. A third party, known as a KDC (Kerberos Key Distribution Center), authenticates the credentials. After authentication, the user may request a service ticket to access one or more services on the network. The ticket includes the encrypted, authenticated identity of the user. The tickets are obtained using the current user's credentials.

VuGen supports tokens based on both Kerberos and Kerberos2 security tokens. The primary difference between the Kerberos and Kerberos2 tokens is that Kerberos2 uses the Security Support Provider Interface (SSPI), so it does not require elevated privileges to impersonate the client's identity. In addition, the Kerberos2 security token can be used to secure SOAP messages sent to a Web Service running in a Web farm.

When you add a Kerberos token to the Vuser script, you specify a logical **Token Name** for the token along with the **Host** and **Domain** names of the Web Services machine.

- **PFX File Token.** These Personal Information Exchange token files (with **.pfx** or **.p12** extensions) contain all of the token information, including the server certificate, intermediate certificates, and the private key in a single file. This file uses the PKCS#12 (Personal Information Exchange Syntax) standard. Click the Browse button to locate the file.



Tip: If you have a JKS keystore (**.jks** file), VuGen automatically converts it to a **.p12** file.

For details about the token attribute in the script, see the Function Reference (**Help > Function Reference**).

Adding the Security Policy

To add a security policy to a section of your script, you enclose the relevant steps with **Web Service Set Security** and **Web Service Cancel Security** steps.

When you add a **Web Services Set Security** step to your script, VuGen adds a **web_service_set_security** function that contains arguments with the tokens, message signatures, and encryption that you defined.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=USERNAME", "TokenName=mytoken1",    "UserName=bob",  
    "Password=123", "PasswordOptions=SendNone",    "Add=True", LAST);
```

Parameterization is not supported for the following arguments: **Token Type**, **Logical Name**, **Base Token**, **Issuer Token** or **Derive From** arguments.

Working with Message Signatures and Encrypted Data

When you add a security token to a SOAP message, it is added to the SOAP message in the form of an XML element in the WS-Security SOAP header.

The message, however, is exposed and therefore requires additional security. This is especially true when the credentials, including the password, are sent in plain text as it is with role-based security.

The two methods used to secure the data are digital signatures and encryption.

- **Digital Signatures.** Digital Signatures are used by message recipients to verify that messages were not altered since their signing. The digital signature is usually in the form of XML within the SOAP message. The recipient checks the signature to make sure it is valid. Certain environments, such as WSE, automatically verify the signature on the SOAP recipient's computer.
- **Encryption.** Although the XML digital signature offers a mechanism for verifying that the message has not been altered since it was signed, it does not encrypt the SOAP message—the message is still plain text in XML format. To secure the message in order that it should not be exposed, you encrypt it, making it difficult for an intruder to view and obtain a user's password.

Parameterization is not supported for message signatures and encryption arguments. For details on adding message signatures and encryption to your script, see ["Add Security to a Web Service Script" on page 801](#).

SAML Security Options

VuGen supports SAML (Security Assertion Markup Language) for Web Services. SAML is an XML standard for exchanging security-related information, called **assertions**, between business partners over the Internet. The assertions can include attribute statements, authentication, decision statements, and authorization decision statements.

SAML uses brokered authentication with a security token issued by STS (Security Token Service). The STS is trusted by the client and the Web Service to provide interoperable security tokens. SAML tokens are important for Web Service security because they provide cross-platform interoperability and a means of exchanging information between clients and services that do not reside within a single security domain.

You can set the SAML settings for an entire script or part of the script. For details, see ["Add SAML Security" on page 804](#).



Note: You cannot apply SAML security and the standard Web Service (a **Web Service Set Security** step) security to the same step. To cancel Web Service security, insert a **Web Service Cancel Security** step.

Signing SAML Assertions

VuGen provides a method for signing an unsigned SAML assertion. As input, you provide the unsigned assertion, a certificate file, and the optional password. As output, VuGen provides the signed SAML

assertion. For task details, see ["Add SAML Security" on page 804](#).

Policy Files

SAML policy files follow the WSE 3.0 standard and define the attribute values for the SAML security. By default, VuGen uses the **samlPolicy.config** file located in the installation's **dat** folder.

When entering SAML security information, you can enter it manually in the properties dialog box, or you can refer to a policy file containing all of the security information. You can create your own policy file based on **samlPolicy.config**.

You can modify the policy file to include values for the security parameters, such as username and certificate information. When adding a SAML security step to your script, if you explicitly specify values for the security arguments, they override the values in the policy file.

If you make changes to the default policy file, we recommend that you copy the new policy file to your script's folder. Make sure to save custom policy files with a **.config** extension to insure that they remain with the script, even when running it on other machines or calling it from the LoadRunner Controller.

To learn more about the SAML policy files, see the SAML STS example on the MSDN Web site. If you want to emulate SAML Federation behavior, copy the **samlFederationPolicy.config** file from the data folder to your script's folder, and specify it as the policy file.

Security Scenarios Overview

VuGen allows you to test Web Services that utilize advanced security and WS-Specifications. Such services can be written in various platforms such as WCF (Windows Communication Foundation), Metro (WSIT), and Axis2. For WCF services, VuGen also supports proprietary standards and transports.

You enable this support by setting up a security scenario. Each scenario represents a typical environment used in conjunction with Web Service calls. VuGen provides several built-in security scenarios that are commonly used. It applies the scenario's settings individually to each service.

For the built-in scenarios, the user interface lets you provide identity information where required. You can customize security, transport, proxy, and other advanced settings.

If you cannot find a scenario that corresponds to your environment, you can use the generic custom scenario.



See also:

- ["Security Scenario Editor Dialog Box" on page 810](#)

Choosing a Security Model

VuGen supports two models for configuring security for your Web Service calls: *Legacy* (no scenario) and *Scenario*. This chapter describes the Scenario security models. The Legacy model refers to the manual addition of Web Service Set Security steps, or the **web_service_set_security** function.

The following table lists the considerations for using each of the models.

Legacy Model	Scenario Based Model
You are working with a script that already uses the legacy model	You are testing a WCF Service
You are testing a service written in frameworks such as .NET 2.0, Axis, or other older toolkits	You are testing a service written in a new framework such as Axis2 or Metro (WSIT).
You require a low-level control over WS-Security tokens	Your service uses advanced specifications such as WS-SecureConversation or WS-Trust
You are having trouble using the new model or find the capabilities of the legacy functions adequate	You are having trouble using the legacy model or you find the capabilities of the new model more adequate

Private, Imported, and Shared Scenarios

To assign a security scenario to a specific service, use the Manage Services window. The **Protocol and Security** tab contains the interface to create and view security scenarios for individual services.

You can select a scenario in three ways:

- **Private scenario.** Create a new scenario by selecting one of the built-in ones and customizing it for your Web Service.
- **Imported scenario.** Use a scenario created at an earlier time. The scenario will be editable, and if someone modifies the original scenario, it will not affect you.
- **Shared scenario.** Load a security scenario already configured by another user from a remote location or the file system. You cannot edit this scenario's settings from the Manage Services window. If someone edits the scenario, it will affect your environment. You usually use this option after working with the product for some time and saving the scenario files.

Scenario Categories

The scenario describes the configuration of your Web Service. It contains information such as security, encoding, proxy, and so forth. VuGen provides a Security Scenario editor that allows you to configure the settings for each scenario.

To determine the scenario that best fits your service, refer to the sections below. If you are unsure which scenario to choose, we recommend to use the **Custom Binding** scenario. For more information, see ["The Custom Binding Scenarios" on page 797](#).

Use the default **<no scenario>** for:

- simple Web Services where no advanced standards are required.
- scripts that use the legacy security model
- Web Services that require a specific security setting, not available in any of the existing scenarios.

If you select a built-in scenario and experience problems in replay, it is possible that no scenario was required and the problem is elsewhere. Reset the value to **<no scenario>**.

The built-in security scenarios are divided into the following categories:

Core Scenarios

The following table describes the built-in Core scenario.

Scenario Name	When to use
Plain SOAP	<ul style="list-style-type: none">• Web services which do not require advanced standards• Web services which may require you to specify the WS-Addressing version

For this type of scenario, if your service uses WS-Addressing, specify the version.

Security Scenarios

The following table describes the built-in Security scenario.

Scenario Name	When to use
Username Authentication	<ul style="list-style-type: none">• Client is authenticated with a username and password on the message level

For this type of scenario, specify the username/password, and if your service uses WS-Addressing, specify the version.

WCF Scenarios

The following table shows the scenarios for Web Services that utilize WCF. The WSHttpBinding-based scenarios are divided according to the way the client authenticates itself to the server. For example, if your client presents a user name and a password to the server, choose the **Username (message protection)** scenario. The user interface lets you provide the identity information in the form of a user name or a certificate as required.

WCF Scenario Name	When to use
WSHttpBinding - No Authentication	<ul style="list-style-type: none">• Client uses the server's X.509 certificate for encryption• Client is not authenticated• Communication may utilize advanced standards such as secure conversation and MTOM

WSHttpBinding - Windows authentication	<ul style="list-style-type: none"> • Client and server use Windows authentication • Security is based on Kerberos or SPNEGO negotiations • Communication may utilize advanced standards such as secure conversation and MTOM
wsHttpBlnding - Certificate authentication	<ul style="list-style-type: none"> • Client uses the server's X.509 certificate for encryption • Client uses its own X.509 certificate for signature • Communication may utilize advanced standards such as secure conversation and MTOM
WSHttpBinding - username (message protection) authentication	<ul style="list-style-type: none"> • Client uses the server's X.509 certificate for encryption • Client is authenticated with a username and password • Communication may utilize advanced standards such as secure conversation and MTOM
WSHttpBinding - username (transport protection) authentication	<ul style="list-style-type: none"> • SSL is enabled • Client is authenticated with a username and password • Communication may utilize advanced standards such as secure conversation and MTOM
WSFederationHttpBinding	<ul style="list-style-type: none"> • Client authenticates against the STS using a predefined scenario • Client uses the token given from the STS to authenticate against the server
Custom Binding	<ul style="list-style-type: none"> • Web Service that uses WS-* standards • WCF services of any configuration

Optimization Scenarios

The following table describes the built-in Optimization scenario.

Scenario Name	When to use
MTOM	<ul style="list-style-type: none"> • MTOM enabled Web services • Web Services which may require you to specify the WS-Addressing version

For MTOM type scenarios, if your service uses WS-Addressing, specify the version.

WCF Scenario Settings

This section describes the values required for the WCF security scenarios:

The WsHttpBinding Scenario

No Authentication (Anonymous)

In this scenario, the client uses the server's certificate to encrypt a message; there is no client authentication.

You specify only one of the following settings:

- **Negotiate service credentials.** Negotiate the Web Service's certificate with the server.
- **Specify service certificate.** Browse for a service certificate. For more information, see ["Select Certificate Dialog Box" on page 815](#). If you select this option, the **Negotiate service credentials** option is not available.

Provide the DNS information.

- **Expected server DNS.** The expected identity of the server in terms of its DNS. This can be **localhost**, an IP address, or a server name. It can also be the common name by which the certificate was issued.

Windows Authentication

This WCF scenario uses Windows Authentication.

You declare the expected identity of the server in terms of its **SPN** or **UPN** identities. If you are testing a WCF service that has not been customized and uses the default configuration, use this type of scenario.

Certificate Authentication

In this WCF WsHttpBinding scenario, the client uses the server's X.509 certificate to encrypt the message and its own certificate for a signature.

Specify only one of the following settings:

- **Negotiate service credentials.** Negotiate the Web Service's certificate with the server.
- **Specify service certificate.** Browse for a service certificate. For details, see ["Select Certificate Dialog Box" on page 815](#). If you select this option, the **Negotiate service credentials** option is not available.

Provide the DNS information:

- **Expected server DNS.** The expected identity of the server in terms of its DNS. This can be **localhost**, an IP address, or a server name. It can also be the common name by which the certificate was issued.

Username Authentication (Message Protection)

In this WCF WsHttpBinding scenario, the client uses the server's X.509 certificate to encrypt the message, and sends a user name and password to authenticate itself.

Specify the following settings:

- **Username. Password.** The client's user name and password credentials.

Specify only one of the following settings:

- **Negotiate service credentials.** Negotiate the Web Service's certificate with the server.

- **Specify service certificate.** Browse for a service certificate. For details, see ["Select Certificate Dialog Box" on page 815](#). If you select this option, the **Negotiate service credentials** option is not available.

Provide the DNS information:

- **Expected server DNS.** The expected identity of the server in terms of its DNS. This can be **localhost**, an IP address, or a server name. It can also be the common name by which the certificate was issued.

Username (Transport Protection) Authentication

This WCF WSHttpBinding scenario enables SSL and authenticates the client with a user name and password on the message level.

Specify the following settings:

- **Username. Password.** The client's user name and password credentials.

The Federation Scenario

In the **WSFederationHttpBinding** scenario, the client authenticates against the STS (Security Token Service) to obtain a token. The client uses the token to authenticate against the application server.

Therefore, two bindings are needed, one against the STS and another against the application server.

First, use the Security Scenario editor to define an STS binding. For more information, see ["Create and Manage Security Scenarios" on page 804](#). When setting the binding against the application server, specify this file in the **Referenced file** box.

For the Federation scenario, specify the following server information:

- **Transport.** HTTP or HTTPS
- **Encoding.** Text or MTOM

For the Federation scenario, specify the following security information:

- **Authentication mode.** IssuedToken, IssuedTokenForCertificate, IssuedTokenForSslNegotiated, IssuedTokenOverTransport, or SecureConversation
- **Bootstrap policy.** IssuedToken, IssuedTokenForCertificate, IssuedTokenForSslNegotiated, or IssuedTokenOverTransport

For the Federation scenario, specify the following identity information:

- **Server certificate.** Browse for a server certificate. For more information, see the ["Select Certificate Dialog Box" on page 815](#).
- **Expected server DNS.** the expected identity of the server in terms of its DNS. This can be **localhost** or an IP address or server name.

For the Federation scenario, specify the following STS (Security Token Service) information:

- **Issuer address.** The address of the issuer of the STS. This can be **localhost**, an IP address, or a server name.
- **Referenced binding.** The file that references the binding that contacts the STS (Security Token Service)

The Custom Binding Scenarios

The **Custom Binding** scenario enables the highest degree of customization. Since it is based upon WCF **customBinding**, it allows you to test most WCF services, along with services on other platforms such as Java that use WS - *<spec_name>* specifications.

Use the **Custom Binding** scenario to configure a custom scenario that does not comply with any of the predefined security scenarios.

For the Custom Binding scenario, specify the following server information:

- **Transport.** HTTP, HTTPS, TCP, or NamedPipe
- **Encoding.** Text, MTOM, or WCF Binary

Specify the following security information:

- **Authentication mode.** None, AnonymousForCertificate, AnonymousForSslNegotiated, CertificateOverTransport, Kerberos, KerberosOverTransport, MutualCertificate, MutualSslNegotiated, SecureConversation, SspiNegotiated, UserNameForCertificate, UserNameForSslNegotiated, UserNameOverTransport, or SspiNegotiatedOverTransport
- **Bootstrap policy.** For SecureConversation type authentication, specify a bootstrap policy: AnonymousForCertificate, AnonymousForSslNegotiated, CertificateOverTransport, Kerberos, KerberosOverTransport, MutualCertificate, MutualSslNegotiated, SspiNegotiated, UserNameForCertificate, UserNameForSslNegotiated, UserNameOverTransport, or SspiNegotiatedOverTransport
- **Net security.** the network security. Select None, Windows stream security, or SSL stream security. For services with HTTP transport, leave the default value, **None**. To enable SSL for HTTP, choose the HTTPS transport.

If your Web Service uses **Reliable messaging**, enable the option, and select **Ordered** or **Not Ordered**.

Identities

Your security settings may require you to provide identity details for either the client and server, or both of them.

An example of identity details for the client, are user name/password or an **X.509** certificate.

For identity information, provide one or more authentication details as required by the service:

Username, Password, Server certificate, Client certificate, or a custom Windows identity. For details about choosing a certificate, see ["Select Certificate Dialog Box" on page 815](#).

Some scenarios require you to declare the expected identity of the server in terms of its DNS, SPN, or UPN identity.

- **DNS.** Provide the name of a server or use localhost.
- **SPN.** Provide the SPN identity in the domain\machine format.
- **UPN.** Provide the UPN identity in the user@domain format.

After setting the basic values, you can set advanced attributes as described in ["Advanced Settings Dialog Box" on page 811](#).

WCF Extensibility

You can implement your own binding, behavior, or channel when using customBinding by defining the assemblyPath and typeName by modifying the configuration file <script directory>/WSDL/@config/[your config].stss.

The assemblyPath attribute should have a value of either the full path of the dll or its relative path to script directory.

The typeName attribute should have the full type name: ns.typeName.

Binding

Name the scenario attribute in the protocols element and provide the assemblyPath and typeName attributes.

The class you use for binding is inherited from System.ServiceModel.Channels.Binding.

Channel

Add a new element under the customization node. You can specify any name for the element, however the element must contain the two attributes:assemblyPath andtypeName.

The class to use for binding is inherited from System.ServiceModel.Channel.BindingElement.

Note: This will work with customBinding scenarios only.

Behavior

Add a new element under the behaviors element (which is under endpointBehavior) and add the two attributes assemblyPath and typeName.

To bind the new element, implement the System.ServiceModel.Description.IEndpointBehavior class.

Note: If you inherit from System.ServiceModel.Description.ClientCredentials, the client credentials from this class will be used.

Examples of Channel and Behavior

```
<protocols scenario="customBinding" uiType="customBinding"
xmlns="http://hp/ServiceTest/config">
  <mode>Private</mode>
  <customization>
```

```
<textMessageEncoding />
<preferlrhttpTransport />
<myChannel assemblyPath="CustomChannel.dll" typeName="CustomChannel.WCFChannel" />
</customization>
<behaviors>
<endpointBehaviors>
<behavior>
<clientVia viaUri="qwqwq" />
<myBehavior assemblyPath="CustomBehavior.dll" typeName="CustomBehavior.WCFbeahvior" />
</behavior>
</endpointBehaviors>
</behaviors>
</protocols>
```

An example of overriding the whole binding (the configuration may contain just one line):

```
<protocols scenario="userBinding" assemblyPath="WCFBinding.dll" typeName="
WCFBinding.Binding"/>
```

Preparing Security Scenarios for Running

Parameterizing Security Elements

You can parameterize the security elements in a script independently. For example, in a username-based security scenario, you might want each Vuser or iteration to use a different user name.

Protecting Custom Headers

When an operation uses SOAP headers, VuGen does not automatically sign or encrypt them. To incorporate a protection scheme such as a signature or encryption, you must manually add the following information to the scenario's configuration file (.stss) in the **behavior** element:

- soapAction of the relevant operation
- The header XML name and namespace
- The protection level

The following example shows an outgoing message with the soapAction, **http://mySoapAction**. The XML element **header1** from namespace **http://myServiceNamespace** is encrypted and signed. The **header2** element from the same namespace is only signed.

```
<protocols ...>
  ...
  <behaviors>
    <contractBehaviors>
      <behavior>
        <channelProtectionBehavior>
          <protectionRequirements action="http://mySoapAction">
            <incomingEncryptionParts>
              <header localName="header1"
namespace="http://myServiceNamespace" />
            </incomingEncryptionParts>

            <incomingSignatureParts>
              <header localName="header1" namespace="
http://myServiceNamespace " />
              <header localName="header2" namespace="
http://myServiceNamespace " />
            </incomingSignatureParts>

          </protectionRequirements>

        </channelProtectionBehavior>
      </behavior>
    </contractBehaviors>
  </behaviors>
</protocols>
```

Emulating Users with Iterations

Many of the security scenarios establish a session with the server. For example, every scenario that uses **WS-SecureConversation** establishes a server session. This session is established when the first operation is executed and ends when the script is finished. By default, VuGen closes all sessions after each iteration and opens them again when the next iteration begins. This implies that every iteration simulates a new session and Vuser.

When working with multiple iterations, this may not be the desired effect—you may prefer to keep the original session active and not open a new session for each iteration. This applies when load testing through the LoadRunner Controller or when setting multiple iterations in the runtime settings.

You can override this behavior so that only the first iteration will establish a new session, while all subsequent ones will continue to use the open session. This simulates a user who repeatedly performs an action using the same session.

To determine which simulation mode to use, choose the one which is most appropriate to what you are simulating. For example, if you are simulating a load test where most of the actions are performed repeatedly by the same user in a single session, use the above configuration. If you are unsure, leave the default settings.

Add Security to a Web Service Script

This task describes how to add set the security for your Web Service calls. For details about Web Services security, see ["Setting Security Overview" on page 787](#).

1. Insert a new Web Services Security Step

- a. Place the cursor at the point at which you want to add the security settings. In most cases, it is best to place it in **vuser_init** so that the security scope will be applied to the whole script. To apply the security for specific calls, place it at the desired location.
- b. Right-click and select **Insert > New Step**. In the Steps Toolbox, under the **Web Services** functions, double-click **web_service_set_security**. The Set Security Properties box opens.

2. Add a security token - optional

- a. Click **Security Tokens** and select a token type. For details, see ["Security Tokens and Encryption" on page 788](#).
- b. In the Token Name box, assign a name for the token to be used by VuGen in identifying the token.
Add any relevant information, such as **User Name** and **Password** for the UserName token type.

3. Add a message signature or encryption - optional

- a. On the Set Security Properties box toolbar, click **Add Message Signature** or **Add Message Encryption**.
- b. In the **Signing/Encryption token** box, select a token to use with the message signature or encryption. Both signatures and encryptions require you to specify a token previously defined as the signing/encrypting token. See ["Add a security token - optional" above](#) above.
- c. Under **What to sign**, specify a target token, or leave the field blank to apply the signature or encryption to the whole message body. For details, see ["Security Tokens and Encryption" on page 788](#).

4. Cancel the security settings - optional

To cancel the security settings at a specific point within the script, add a **web_service_cancel_security** step at the desired point.



See also:

- ["Set Security Properties Dialog Box" on page 807](#)
- ["Customize the Security" below](#)
- ["Add SAML Security" on page 804](#)

Customize the Security

This task describes how to how to configure special cases common to Web Service security.

Reference a token with a SubjectKeyIdentifier - optional

By default, Service Test adds all of the defined X.509 tokens to the SOAP envelope and references

them as binary tokens. It is also possible to exclude the tokens from the message and reference them with a SKI (Subject Key Identifier). This is common with tokens that are used for encryption.

1. Add a token as described in the ["Add Security to a Web Service Script" on the previous page](#).
2. In the script, change the value for **Add** to **false**:

```
SECURITY_TOKEN, "Type=X509", "LogicalName=myToken", "StoreName=My",  
"IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser",  
"Add=False",
```

3. If necessary, set the **useRFC3280** settings as described in ["useRFC3280" on page 804](#) below.

You can customize the Username token with a nonce and timestamp.

1. Locate the **web_service_set_security** function in the script.
2. Add the attributes and their values according to this chart:

Name	Meaning	Possible values
IsNonceIncluded	Include a nonce with the token.	True (default) or False
TimestampFormat	The timestamp format to use with the token.	<ul style="list-style-type: none">• None. no timestamp• Full. a <timestamp> element with <created> and <expired> inner elements• Created. (default) only a <created> element

For example:

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=USERNAME", "LogicalName=myToken", "UserName=John",  
    "Password=1234", "PasswordOptions=SendPlainText", "IsNonceIncluded=true",  
    "TimestampFormat=Full", "Add=True",  
    LAST);
```

Customize the encryption - optional

You customize encryption by indicating whether to encrypt the whole element or only its content. This is common when encrypting tokens such as a user name. By default, only the content is encrypted. The following steps describe how to encrypt the entire token.

1. Locate the **web_service_set_security** function in the script.
2. Add the **EncryptionType** attribute with the value **Element**.

```
web_service_set_security(  

```

```
...
ENCRYPTED_DATA, "UseToken=myToken", "TargetToken=myOtherToken",
"EncryptionType=Element",
LAST);
```

3. To return to the default, remove the **EncryptionType** attribute or set it to **Content**.

Customize WS-Security - optional

To change the algorithm Service Test uses for encryption or to modify some other low-level security details.

1. To change either of these items, open the **%Service Test%/bin/mmdrv.exe.config** file in a text editor.
2. If this file does not contain the **<microsoft.web.services2>** element, add it as shown below.

```
<configuration>
...
  <microsoft.web.services2>
    <security>
      <x509 storeLocation="CurrentUser" allowTestRoot="true" useRFC3280="true" />

      <binarySecurityTokenManager valueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">
        <sessionKeyAlgorithm name="TripleDES" />
        <keyAlgorithm name="RSA15" />
      </binarySecurityTokenManager>
    </security>
  </microsoft.web.services2>
...
</configuration>
```

3. Set the element values as required:

Name	Meaning	Possible values
verifyTrusy	Check sent/received x.509 certificate's validity.	<ul style="list-style-type: none"> • True (default) • False
sessionKeyAlgorithm	The algorithm the session symmetric key should use to encrypt the message.	<ul style="list-style-type: none"> • AES128 • AES192 • AES256 • TripleDES

keyAlgorithm	The algorithm to use by the public key to encrypt the session key.	<ul style="list-style-type: none">• RSA15• RSASOAE P
useRFC3280	Generate subject key identifiers that are interoperable and not Windows specific.	<ul style="list-style-type: none">• True• False (default)

Add SAML Security

This task describes how to add SAML security for your Web Service calls.

1. Insert a new Web Services Security step.
 - a. Place the cursor where you want to add the security settings.
 - b. Right-click and select **Insert > New Step**.
 - c. In Steps Toolbox > SOAP functions, select **web_service_set_security_saml**.
2. Insert a SAML assertion.
 - a. Right-click and select **Insert > New Step**.
 - b. In Steps Toolbox > SOAP functions, select **ws_sign_saml_assertion**.
 - c. Provide the unsigned assertion, a certificate file, and a password. (Optional)
3. Set the security policy. (Optional)

Specify a policy file, or leave blank to use the default. Manually enter values override any values in the policy file. Make sure to provide an Issuer URL, also known as the **STS URL**.
4. Cancel the SAML settings. (Optional)

At the required location in the script, insert a **web_service_cancel_security_saml** step.



See also:

- ["SAML Security Options" on page 790](#)
- For syntax information, see the Function Reference (**Help > Function Reference**)

Create and Manage Security Scenarios

The following steps describes how to create and customize a security scenario for a specific service.

1. Open the Security Scenario Data dialog box.
 - a. Click **Manage Services**. In the left pane, select the service for which you want to set the security scenario. If necessary, import a service, as described in ["Import Service Dialog Box" on page 784](#).
 - b. Select the **Protocol and Security** tab and click the **Edit Data** button. The Security Scenario Data dialog box opens.

2. Create or open a security scenario.

Create a security scenario

- a. Select **Private scenario** and select a built-in security scenario for the current service.
- b. In the **Scenario type** box, select a scenario. For details, see ["Choosing a Security Model" on page 791.](#)
- c. Specify the required values for your scenario. For details, see ["WCF Scenario Settings" on page 794.](#)
- d. To specify a certificate (only applicable to some of the scenarios), click the Browse button adjacent to the **Client certificate** or **Specify service certificate** box to open the Select Certificate dialog box. For details, see the ["Select Certificate Dialog Box" on page 815.](#)
 - **To retrieve a certificate from a file:** Select **File** and locate its path.
 - **To retrieve a certificate from the Windows store:** Select **Windows Store**. Then select a store location and name, and specify a search string:
 - To search for all certificates, leave the **Search text** box empty.
 - To search for a specific certificate, specify a substring of the certificate name. If required, specify a password for the private key.Click **Find** to generate the list of certificates found in the store.

Open a security scenario

- **To use an editable scenario:** Select **Private scenario** and click **Import**. In the Shared Scenario dialog box, select a stored scenario. If required, modify the settings as described in ["WCF Scenario Settings" on page 794.](#)
- **To use a non-editable scenario:** Select **Shared Scenario**. Then browse to and select a stored scenario.

Note: If someone modifies a shared scenario file at its source, it will affect your script.

3. Modify the security scenario, as described below. (Optional)

To:	Do this:
Configure advanced settings	<p>For most scenarios, the default proxy, encoding, and other settings are ideal, but you can modify these settings if needed.</p> <ol style="list-style-type: none">a. Click Advanced to configure the settings. For details, see "Advanced Settings Dialog Box" on page 811.b. Click OK to save the security scenario.

To:	Do this:
Modify an existing security scenario	<p>To create and modify security scenarios that will be available globally for all scripts—not just this specific service, use the Security Scenario Editor. You can also use the editor to save the scenario.</p> <ol style="list-style-type: none"> Select SOA Tools > Security Scenario Editor. Click the Load button and browse for an existing stss scenario file. Modify the scenario settings as required Click Save or Save as.
Protect SOAP headers	<p>Manually modify the behavior element in the scenario's configuration file:</p> <ol style="list-style-type: none"> In VuGen, open the Script view. Choose View > Script View. Click in the script editor and select Open Script Directory from the shortcut menu. Locate the security scenario's configuration file <code><service_name>.stss</code> in WSDL/@config folder. Modify the behavior section of the file. For details, see "Protecting Custom Headers" on page 799.
Set the iteration mode	<p>Configure your environment to use the same session for all iterations:</p> <ol style="list-style-type: none"> Open the script root folder: In Script view, click inside the script and choose Open Script Directory from the shortcut menu. Open default.cfg file in a text editor. In the [WebServices] section, add in a row under the toolkit. If you are using the Axis toolkit or if you configured other settings, the file contents may differ. <pre> [WebServices] Toolkit=.NET SimulateNewUserInNewIteration=0 </pre> Save and close the file. <p>For details, see "Emulating Users with Iterations" on page 800.</p>

Parameterize Security Elements

This task describes how to independently parameterize the security elements in a script.

1. Open the Security Scenario Editor

Select **SOA Tools > Security Scenario Editor**.

2. Set up a scenario for each Vuser

Set up a scenario for each Vuser as described in ["Create and Manage Security Scenarios" on page](#)

804. We recommend you use the names **user1**, **user2**, and so forth, and save them in a new folder, **%script root%/WSDL/referencedConfig**.

3. Open the Parameter List window and create a parameter

Select **Vuser > Parameters List**. Create a new parameter, **<ServiceName>_shared_config**. Replace the **<ServiceName>** with the case-sensitive name of the service you are testing. To determine the exact name of the service, click **Manage Services** to see the list of services.

4. Add parameter values

In the values table, in each row add the file names of the security scenarios with their .stss extensions. You can use a relative path, relative to the script folder. Click **Add Row** to add multiple values. Close the Parameter List dialog box.

5. Call the parameter

- Click **Manage Services** and select the **Protocol and Security** tab. Click **Edit Data**.
- Select **Shared Scenario**. Click the Browse button and enter the parameter name, **<ServiceName>_shared_config**, in the test box.

Set Security Properties Dialog Box

This dialog box enables you to set the security properties for your Web Service calls.

To access	In the Steps Toolbox pane, double-click the web_service_set_security step.
Relevant tasks	"Add Security to a Web Service Script" on page 801
Important Information	If you have edited key algorithm or session algorithm values in the mmdrv.config file for an existing script, these values are replaced with the system default values.

WS—Security Tab

User interface elements are described below (unlabeled elements are shown in angle brackets):

UI Element	Description
<Token Grid>	Displays a unique number, type and name of all tokens that have been added.



Add

Security Token

Enables you to select a token type:

UserName Token

- **Token Name.** A meaningful name for the token.
- **Include nonce.** If selected, an arbitrary number is used once to sign communication.
- **Username/Password.** Specify the username/password.
- **Password type.** Text, Hash, None
- **Timestamp format.** Full, Created, None

X.509 Certificate Token




- **Token Name.** A meaningful name for the token.
- **Certificate.** If selected, an arbitrary number is used once to sign communication.
- **Reference Type.** BinarySecurityToken, Reference

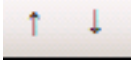
Kerberos/Kerberos2 Token

- **Token Name.** A meaningful name for the token.
- **Host.** The host name of the server against which you want to authenticate. In most cases, it is the host portion of the service URL.
- **Domain.** The Windows domain of the server against which you want to authenticate.

PFX File Token

- **Token Name.** A meaningful name for the token.
- **PFX File.** Contains all of the token information, including the server certificate, intermediate certificates, and the private key in a single file.
Tip: If you have a **.jks** file, use it here. VuGen will automatically convert it to a **.p12** file.
- **PFX File Password.** The password for your PFX file.
Note: If you are using a **.jks** file, use your .jks file password.

 <p>Add Message Signature</p>	<ul style="list-style-type: none"> • Signing token. The token to use for signing, usually an X.509 type. Select from the list of all added tokens. • Canonicalization algorithm. A URL for the algorithm to use for canonicalization. A drop-down list provides common algorithms. If you are unsure which value to use, keep the default. • Transform Algorithm. A URL for the Transform algorithm to apply to the message signature. A dropdown list provides common algorithms. If you are unsure which value to use, keep the default. • Inclusive namespace list. A list of comma-separated prefixes to be treated as inclusive (optional). • What to sign. The SOAP elements to sign: SOAP Body, Timestamp, and WS-Addressing. <ul style="list-style-type: none"> • Xpath (optional). An XPath that specifies which parts in the message to sign. If left blank, the elements selected in the Signature options field are signed. For example, <code>//*[local-name(.)='Body']</code>. • Token (optional). The target token you want to sign. Select from the drop-down list of all added tokens. With most services, this field should be left empty.
 <p>Add Message Encryption</p>	<ul style="list-style-type: none"> • Encrypting Token. The token to use for encryption, usually an X.509 type. You can select from a list of all previously created tokens. • Encrypting Type. Indicates whether to encrypt the whole destination Element or only its Content. • Key algorithm. The algorithm to use for the encryption of the session key: RSA15 or RSAOAEP. If you have edited the mmdrv.config file with a custom key algorithm value for an existing script, this value is replaced with the system default value of RSA15. • Session algorithm. The algorithm to use for the encryption of the SOAP message. You can select from a list of common values. If you have edited the mmdrv.config file with a custom session algorithm value for an existing script, this value is replaced with the system default of AES128. • What to encrypt. <ul style="list-style-type: none"> • Xpath (optional). An XPath that indicates the parts of the message to encrypt. If left blank, only the SOAP body is encrypted. • Token (optional). The name of the encrypted token. A drop-down box provides a list of all added tokens. With most services, this field should be left empty.
	<p>Delete a token definition from the grid.</p>

	<p>Up/Down. Positioning tools that allow you to set the priority of the security elements.</p> <div data-bbox="407 331 1412 411"> <p>Note: Make sure the security elements are positioned in order of their priority.</p> </div>
<p>Exclude Timestamp</p>	<p>Removes the timestamp from the SOAP header before sending the security element to the server.</p>

WS Addressing

The WS-Addressing tab indicates whether WS-Addressing is used by the service, and if so, its version number. You can also specify the IP address of the server to which you want the response to be sent.

See also:







- ["Security Tokens and Encryption" on page 788](#)
- ["Add SAML Security" on page 804](#)

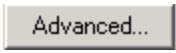
Security Scenario Editor Dialog Box

This dialog box enables you to define security scenarios for your script.

To access	SOA Tools > Security Scenario Editor
Important information	You can also define scenarios for a specific service. For details, see "Create and Manage Security Scenarios" on page 804 .

User interface elements are described below:

UI Element	Description
	<p>New. Resets the editor for defining a new security scenario. If you made changes to the current scenario, it prompts you to save them.</p>
	<p>Load. Opens an existing shared scenario from a URL or file.</p>
	<p>Save. Saves the scenario file. If you have not saved the file at least once, it prompts you for a name.</p>
	<p>Save as. Saves the scenario file at a new location.</p>
	<p>Help. Opens the Online help for security scenarios.</p>
	<p>Close. Closes the dialog box.</p>

	Opens the Advanced Setting dialog box for setting the encoding, reliable messaging, secure session information, and proxy configuration. For details, see "Advanced Settings Dialog Box" below .
Scenario type	The security scenario type: No scenario or a sub-type of Core, Security, WCF, or Optimization scenarios.

Advanced Settings Dialog Box

This dialog box lets you configure advanced settings for security scenario in the areas of Encoding, Advanced Standards, Security, or HTTP and Proxy. You access these setting via the ["Security Scenario Editor Dialog Box" on the previous page](#).

Not all settings are relevant for all scenarios—some of them might be disabled or hidden depending on the scenario type.

Encoding

The Encoding tab lets you indicate the type of encoding to use for the messages: **Text**, **MTOM**, or **Binary**. The default is **Text** encoding.

For each of these encoding methods, you can choose a version of WS-Addressing:

- None
- WSA 1.0
- WSA 04/08



Tip: To instruct a SOAP request to leave out WS-Addressing, add a **web_service_set_option** function with the **ExcludeWseHeaders** flag set to *true* before the **web_service_call** function. For details, see the [Function Reference \(Help > Function Reference\)](#).

Advanced Standards

This tab lets you configure advanced WS- standards, such as Reliable Messaging and the Via address option.

If your service implements the **WS-ReliableMessaging** specification, enable the **Reliable Messaging** option and set the following options:

- **Reliable messaging ordered.** indicates whether the reliable session should be ordered
- **Reliable messaging version.** WSReliableMessagingFebruary2005 or WSReliableMessaging11

Via Address

In certain instances, you may need to send a message to an intermediate service that submits it to the actual server. This may also apply when you send the message to a debugging proxy. This corresponds to the WCF **clientVia** behavior.

In such cases it may be useful to separate the physical address to which the message is actually sent, from the logical address for which the message is intended. The logical address may be the physical address of the final server or any name. It appears in the SOAP message as follows:

```
<wsa:Action>http://myLogicalAddress</wsa:Action>
```

The logical address is retrieved from the user interface. By default, it is the address specified in the WSDL. You can override this address from the Manage Services dialog box.

Security

The Advanced security settings correspond to the **WS-Security** specifications.

For security scenarios that are based upon WCF **WSHttpBinding**, you can indicate the following settings:

- **Enable secure session.** Establish a security context using the WS-SecureConversation standard.
- **Negotiate service credentials.** Allow WCF proprietary negotiations to negotiate the service's security.

For **WSHttpBinding**, **Custom Binding**, or **WSFederationHttpBinding** WCF type scenarios, you can set the default algorithm suite and protection level:

Attribute	Meaning	Possible Values
Default Algorithm Suite	The algorithm to use for symmetric/asymmetric encryption. These are the values from the SecurityAlgorithmSuite configuration in WCF:	<ul style="list-style-type: none"> • Basic128 • Basic128Rsa15 • Basic128Sha256 • Basic128Sha256Rsa15 • Basic192 • Basic192Rsa15 • Basic192Sha256 • Basic192Sha256Rsa15 • Basic256 • Basic256Rsa15 • Basic256Sha256 • Basic256Sha256Rsa15 • TripleDes • TripleDesRsa15 • TripleDesSha256 • TripleDesSha256Rsa15
Protection Level	Should the SOAP Body be encrypted/signed	None, Sign, and EncryptAndSign (default)

For **Custom Binding** or **WSFederationHttpBinding** WCF type scenarios, you can customize the security settings in greater detail. The following table describes the options and their values:

Attribute	Meaning	Possible Values
Message Protection Order	The order for signing and encrypting	<ul style="list-style-type: none"> • SignBeforeEncrypt • SignBeforeEncrypt-AndEncryptSignature • EncryptBeforeSign
Message Security Version	The WS-Security security version	A list of the current versions
Security Header Layout	The layout for the message header	<ul style="list-style-type: none"> • Strict • Lax • LaxTimeStampFirst • LaxTimeStampLast
Key Entropy Mode	The entropy mode for the security key.	<ul style="list-style-type: none"> • Client Entropy • Security Entropy • Combined Entropy

You can enable or disable the following options:

- **Require derived keys.** Indicates whether or not to require derived keys.
- **Require security context cancellation.** Disabling this option implies that stateful security tokens will be used in the **WS-SecureConversation** session (if enabled).
- **Include timestamp.** Includes a timestamp in the header.
- **Allow serialized token on reply.** Enables the reply to send a serialized token.
- **Require signature confirmation.** Instructs the server to send a signature confirmation in the response.

For X.509 certificates, you can specify values for the following items:

Attribute	Meaning	Possible Values
X509 Inclusion Mode	When to include the X509 certificate	<ul style="list-style-type: none"> • Always to Recipient • Never • Once • AlwaysToInitiator
X509 Reference Style	How to reference the certificate	<ul style="list-style-type: none"> • Internal • External

X509 require derived keys	Should X509 certificates require derived keys	<ul style="list-style-type: none"> • Enable - Yes • Disable - No
X509 key identifier clause type	The type of clause used to identify the X509 key.	<ul style="list-style-type: none"> • Any • Thumbprint • IssuerSerial • SubjectKeyIdentifier • RawDataKeyIdentifier

HTTP and Proxy

This tab lets you set the HTTP and Proxy information for your test.

HTTP(S) Transport

The following table describes the HTTP(S) Transport options:

Option	Meaning	Possible Values
Transfer mode	The transfer method for requests/responses	Buffered, Streamed, StreamedRequest, StreamedResponse
Max response size (KB)	The maximum size of the response before being concatenated	Default 65 KB
Allow cookies	Enable cookies	Enabled/Disabled
Keep-Alive Enabled	Enable keep-alive connections	Enabled/Disabled
Authentication scheme	HTTP authentication method	None, Digest, Negotiate, NTLM, IntegratedWindows Authentication, Basic, Anonymous
Realm	The realm of the authentication scheme	Any URL
Require client certificate	For SSL transport, require a certificate	Enabled/Disabled

Proxy Information

If the Web service's transport uses a proxy server, you can specify its details in the **Security** tab. The following table describes the proxy options:

Option	Meaning	Possible Values
--------	---------	-----------------

Use default web proxy	Use machine's default proxy settings	Enabled/Disabled
Bypass proxy on local	Ignore proxy when the service is on the local machine	Enabled/Disabled
Proxy address	the proxy server	Any URL
Proxy authentication scheme	HTTP authentication method on Proxy	None, Digest, Negotiate, NTLM, IntegratedWindows Authentication, Basic, Anonymous


Select Certificate Dialog Box

This dialog box enables you to search and locate a certificate from a file or Windows store.

To access	<p>Select a scenario that uses a certificate in one of the following ways:</p> <ul style="list-style-type: none"> • Open the Security Scenario Editor: Choose SOA Tools > Security Scenario Editor. • In the Manage Services dialog box, select the Protocol and Security tab and click the Edit Data button. <p>Select a WCF scenario that uses a client or service certificate, such as WsHttpBinding or WSFederationHttpBinding. In the Certificate field, click the Browse button.</p>
Important information	This only applies to security scenarios that allow you to specify client, server, or service certificates.
Relevant tasks	<ul style="list-style-type: none"> • "Add Security to a Web Service Script" on page 801 • "Create and Manage Security Scenarios" on page 804.

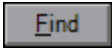



Select Certificate from File

When you choose **File**, the dialog box shows the user interface elements described below:

UI Element	Description
	Browse. Allows you to locate the certificate file with a .pem, .arm, .der, or .pfx extension.
File	The complete path of the certificate file.
Password (optional)	The password required to access the certificate.

Select Certificate from Windows Store

When you choose **Windows Store**, the dialog box shows the user interface elements described below (unlabeled elements are shown in angle brackets>):

UI Element	Description
	Begins the search for the certificate.
Import from	The location of the certificate: <ul style="list-style-type: none"> • Windows store • File
Store location	The store location, for example Current User .
Store name	The store name, for example, AuthRoot .
Search text	<p>The text to match in the certificate name.</p> <p>By default, the search mechanism searches for the text by the certificate's <i>subject name</i>.</p> <div>  <p>Tip: To search for an X.509 certificate by its <i>serial number</i> or <i>issuer name</i>, drag a web_service_set_security function from the Steps Toolbox, into your script. From the Security Tokens drop-down , select X509 Certificate Token. Click the Browse button  adjacent to the Certificate field. In the Select Certificate dialog box, select <i>Issuer Name</i> or <i>Serial Number</i> from the Search field drop-down list.</p> </div>
Password (optional)	The password required to access the certificate.
<certificate list>	A list of the certificates in the Windows store sorted by Subject, Issuer, Private, Store Location, and Store Name.

Web Services Security Examples

This section illustrates several common security scenarios.

Authenticating with a Username Token

The following example illustrates the sending of a message level username/password token (a username token), where the user name is John and the password is 1234.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=USERNAME", "LogicalName=myToken", "UserName=John",  
    "Password=1234", "PasswordOptions=SendPlainText", "Add=True",  
    LAST);
```

Signing a Specific Element with an X.509 Certificate

It is possible to sign only a specific element in a message. The following example signs a specific element using an XPATH expression:

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=myCert", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser", "Add=True",  
    MESSAGE_SIGNATURE, "UseToken=myCert", "TargetPath=//*[local-name(.)  
    ='someElement' and namespace-uri()='http://myNamespace']",  
    LAST);
```

Signing with an X.509 Certificate

The following example shows a script using an X.509 certificate for a digital signature.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=myCert", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser", "Add=True",  
    MESSAGE_SIGNATURE, "UseToken=myCert",  
    LAST);
```

Note: The certificate needs to be installed in the Windows certificate store. In the example above, you need to set the actual store name, store location, and subject name of your certificate.

Encrypting with a Certificate

The following sample encrypts a message with the service's X.509 certificate.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=serviceCert", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=serviceCert", "StoreLocation=CurrentUser",  
    "Add=False",  
    ENCRYPTED_DATA, "UseToken=serviceCert",  
    LAST);
```

After you specify the details of your X.509 certificate, you can encrypt a specific XPATH in the message. Since we want to generate a Subject Key Identifier, we set the Add value to **False**.

Authenticating with a Username Token and Encrypting with an X.509 Certificate

The following example sends a username token to the service and encrypts it with the server's X.509 certificate:

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=serviceCert", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=serviceCert", "StoreLocation=CurrentUser",  
    "Add=True",  
    SECURITY_TOKEN, "Type=USERNAME", "LogicalName=myUser", "UserName=John",  
    "Password=1234", "PasswordOptions=SendPlainText", "Add=True",  
    ENCRYPTED_DATA, "UseToken=serviceCert", "TargetToken=myUser",  
    LAST);
```

The **UseToken** and **TargetToken** properties indicate which token to use and which to encrypt. Their values reference the **LogicalName** property of the tokens.

Encrypting and Signing a Message

This example shows how to sign a message using a private key and then encrypt it using the service's public key.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=myCert", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser", "Add=True",  
    SECURITY_TOKEN, "Type=X509", "LogicalName=serverToken", "StoreName=My",  
    "IDType=SubjectName", "IDValue=CN=serverCert", "StoreLocation=CurrentUser",  
    "Add=False",  
    MESSAGE_SIGNATURE, "UseToken=myCert",  
    ENCRYPTED_DATA, "UseToken=serverCert",  
    LAST);
```

Referencing an X.509 Certificate Using a Hash

In certain cases, you may be unable to reference a certificate with a subject name. This example shows how to reference the certificate using its unique hash.

```
web_service_set_security(  
    SECURITY_TOKEN, "Type=X509", "LogicalName=serviceCert", "StoreName=My",  
    "IDType=Base64KeyID", "IDValue=pOIO+1iuotKLIO91nhjDg5reEw0=",  
    "StoreLocation=CurrentUser", "Add=False",  
    ENCRYPTED_DATA, "UseToken=serviceCert",  
    LAST);
```

Troubleshooting and Limitations for Web Services

This section describes troubleshooting and limitations for the Web Services protocol.



Tip: For general VuGen troubleshooting and limitations, see "[Troubleshooting and Limitations for VuGen](#)" on page 868.

- **Issue:** When you run a **web_service_call** function, the function may fail because a security token is included in the response.

Workaround: Use the **web_service_set_option** function to turn on the **DoNotValidateSecurity** option, using the syntax below:

```
web_service_set_option("DoNotValidateSecurity", "true");
```

- If your script contains one of the following API functions: **web_service_call**, **web_service_set_security**, or **web_service_set_security_saml**, you will encounter an error if WSE 2.0 SP3 and WSE 3.0 are not installed.
Solution:
 - First, activate .NET 3.5. For Windows 8.1/2012 R2, turn on the feature as described in the [MSDN](#). For Windows 7/2008 R2, download and install [.NET Framework 3.5](#).
 - Next, install the WSE components from the LoadRunner DVD folders, **lrunner\Common\wse20sp3** and **lrunner\Common\wse30**, or download them from the Internet.
- The **Record default web browser** option in the Recording Wizard, is only supported for Internet Explorer.
- For large SOAP envelopes, Record and Replay snapshots are disabled.
- The Import SOAP feature is not supported for envelopes containing a single element larger than 500KB.
- Recording requests with attachments or security is not supported.
- For Axis toolkit, Web service calls that include both attachments and security are not supported.
- For .NET toolkit, SOAP version 1.2 is not supported for asynchronous calls.
- You can enter text strings up to 10 KB to encode to base 64. If your string is larger, use the **Get from file** option.
- VuGen supports Web Service messages over JMS message Queue, but does not support JMS Topics.
- If the response is MIME format, replay may fail for Web services imported through the .NET toolkit.
Workaround: Insert a **web_service_set_option** function before the **web_service_call**. Enable the **HandleMIMEResponse** attribute:

```
web_service_set_option("HandleMIMEResponse", "true")
```
- JMS Bindings Extensions are not supported.
- All services in your script should have the same security scenario. This can be configured via the Protocols and Security tab.
- Asynchronous Web Service calls and custom user handlers are not supported for WCF.
- LoadRunner cannot replay scripts containing the **soa_xml_validate** function.
- When using **Update service**, steps that are already in the script will not display the updated properties

(in the step argument view) until you close and reopen the application. After you reopen the app, step arguments are updated. If the script is open when performing "update service", then on the script view arguments, the application throws an exception.

Workaround: Close the script file while running "update service", or reopen the test after running "update service".

- A Web Service script might not open when you import the WDSL with the Axis toolkit.**Workaround:** Import the WDSL with the .NET toolkit.

If there is a problem recreating the scripts, do the following:

- a. Create a new test.
- b. Import the WSDL using .NET toolkit.
- c. Go to the directory of the new script.
- d. Copy the folder "WSDL" and paste it on the directory of the old script.
- e. In the directory of the old script open the **default.cfg** file.
- f. Under [WebServices] header, change "**Toolkit=Axis**" to "**Toolkit=.NET**".

Windows Sockets Protocol

Recording Windows Sockets - Overview

The Windows Sockets protocol supports applications which communicate over the TCP/IP protocol using a Microsoft WinSock DLL. The WinSock protocol allows you to see the actual data sent and received by the buffers.

The WinSock protocol records functions that relate to the sockets, data buffers, and the Windows Sockets environment. Using VuGen, you record your application's API calls to the Winsock.dll or Wsock32.dll. For example, you could create a script by recording the actions of a telnet application. After creating a Winsock Vuser script, you can view the recorded buffers as raw data or as a snapshot.



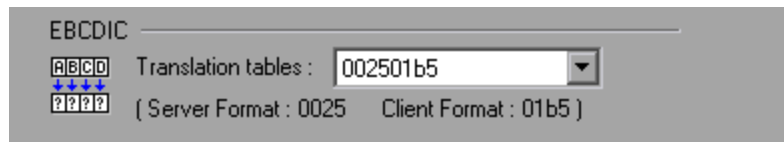
See also:

- ["Windows Sockets Data" on the next page](#)
- ["Windows Sockets Snapshots - Overview" on page 822](#)

Translation Tables

You can display Windows Sockets data in EBCDIC format through a translation table.

A translation table allows you to specify the format for recording when using the WinSock single protocol, and for code generation when using a WinSock multi protocol. This applies to users running on mainframe machines or AS/400 servers. Both the server and client machines determine the format of the data from translation tables installed on your system. If your data is in ASCII format, it does not require translation.



The first four digits of the listbox item represent the server format. The last four digits represent the client format. In the above example, the selected translation table is 002501b5. The server format is 0025 and the client format is 01b5 indicating a transfer from the server to the client. In a transmission from the client to the server, you would select the item that reverses the formats—01b50025 indicating that the client's 01b5 format needs to be translated to the server's 0025 format.

The translation tables are located in the **ebcdic** folder under the VuGen's installation folder. If your system uses different translation tables, copy them to the **ebcdic** folder.

For details on selecting a translation table in the recording options, see the ["WinSock Recording Options" on page 203](#).

Windows Sockets Data

When you use VuGen to create a Windows Sockets Vuser script, your actions are recorded into the three sections of the script: **vuser_init**, **Action**, and **vuser_end**. In addition to the Vuser script, VuGen also creates data files:

- **snapshotdata.ws** contains the data that was transmitted or received during the recording session. VuGen's Snapshot pane displays the contents of the data file. Do not modify the contents of the **snapshotdata.ws** file.
- **data.ws** contains the data that is transmitted during the replay sessions, and is expected to be received. You can right-click any step in the Editor and then select **Show Arguments** to show the buffer content that is stored in **data.ws** for the selected step. Using the **Text View** tab of the dialog box that opens, you can edit the data that is stored for any data buffer.

Several LRS functions, such as **lrs_receive** and **lrs_send**, handle the actual data that is transferred between servers and clients. The data that is received or transmitted is stored in data buffers, which can be very large. In order to simplify the appearance of the Vuser script, the actual data is stored in external files—not in the C file. When a data transfer occurs, the data is copied from the external file into a temporary buffer.

The external file, **data.ws**, contains the contents of all the temporary buffers. The buffers' contents are stored as sequential records. The records are marked by identifiers indicating whether the data was sent or received, and the buffer descriptor. The LRS functions use the buffer descriptors to access the data.

The descriptors have one of the following formats:

recv buf index	number of bytes received
send buf index	number of bytes sent

The buffer index begins with 0 (zero), and all subsequent buffers are numbered sequentially (1,2,3...) regardless of whether they are send or receive buffers.

In the following example, an **lrs_receive** function was recorded during a Vuser session:

```
lrs_receive("socket1", "buf4", LrsLastArg)
```

In this example, **lrs_receive** handled data that was received on socket1. The data was stored in the fifth receive record(buf4)—note that the index number is zero-based. The corresponding section of the **data.ws** file shows the buffer and its contents.

```
recv buf4 39
"\xff\xfb\x01\xff\xfb\x03\xff\xfd\x01"
"\r\n"
"\r\n"
"SunOS UNIX (sunny)\r\n"
"\r"
"\x0"
"\r\n"
"\r"
"\x0"
```

For task details, see ["View and Modify Windows Sockets Buffers" on page 826](#).

Windows Sockets Snapshots - Overview

Vuser scripts based on the Windows Sockets Vuser protocol utilize VuGen's Snapshot pane.

For Windows Sockets Vuser scripts, the Snapshot pane displays snapshots of the recorded data buffers. The following sections describe how to view and use the data.

Displaying Buffer Data

To display a specific buffer in the Snapshot pane:

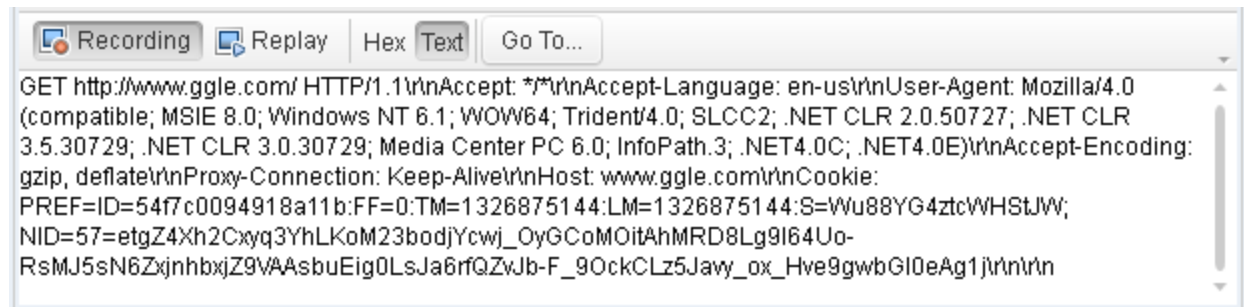
- In the Editor, select the step that contains a reference to the required buffer.
- In the Step Navigator, double-click the step that contains a reference to the required buffer.

Buffer Views

You can view the buffer snapshots in either **Text** view or **Hex** view.

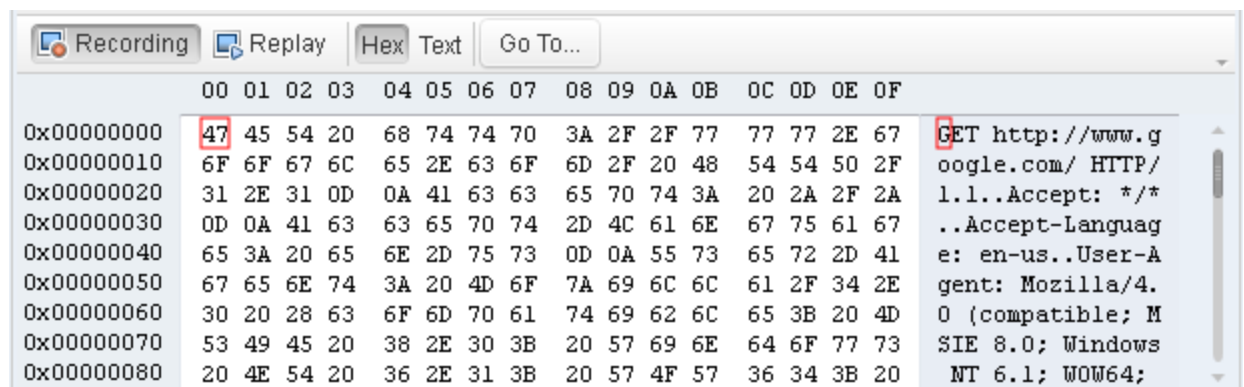
Note: VuGen stores the snapshot data as read-only data. You cannot modify the contents of the snapshots. However, you can modify the buffer data that is associated with any of the steps in a Vuser script. To modify the buffer data, right-click the required step in the Editor and then select **Show Arguments**. The Text View tab of the dialog box that opens lets you modify the buffer data. For details, see ["View and Modify Windows Sockets Buffers" on page 826](#).

The **Text** view shows the buffer data as text.



The **Hex** view shows the buffer data in hexadecimal representation. The data is displayed in three columns:

- The left column shows the offset of the first character in each row.
- The middle column shows the hexadecimal value of the data.
- The right column shows the data in ASCII format.



Status Bar

The status bar below the buffer snapshot displays information about the buffer and the data selected in the buffer:

- **Buffer number.** The buffer number of the displayed buffer.
- **Buffer size.** The total number of bytes in the buffer.
- **Buffer type.** The type of buffer—received or sent.
- **Data.** The value of the data that is selected in the buffer, in decimal and hexadecimal formats. Both big endian and little endian sequences are displayed.
- **Offset.** The offset of the selected data from the beginning of the buffer. If you select multiple bytes, the offset displays the range of the selection.

buf5: 587 bytes(s) received	BE: 1129324658 (0x43502072)	LE: 1914720323 (0x72205043)	Selection: from 235 (0xEB) to 239 (0xEF)
-----------------------------	--------------------------------	--------------------------------	---

Navigating within the Buffer Data

- To go to a specific offset within the buffer (absolute), click **Go To**. In the Go To Offset dialog box enter an offset value, and then click **Apply**.

- To jump to a location relative to the selected entry, click **Go To**. In the Go To Offset dialog box, click **Advance by**, specify the number of bytes to advance, and then click **Apply**.

Creating Correlations from the Buffer Data

Creating Correlations from the Buffer Data

Using the shortcut menu, you can create correlations directly from the **Recording** tab in the Snapshot pane.

- To create a regular correlation, select the data and choose **Create Correlation** from the shortcut menu. VuGen adds an **lrs_save_param** function to the script.
- To create a boundary bounded correlation, select the data and choose **Create Boundary Correlation** from the shortcut menu. VuGen adds an **lrs_save_searched_string** function to the script.

Data Navigation Tools

VuGen provides you with **Go To** functionality to help you to navigate through the data in the Snapshot pane. This helps you to identify and analyze a specific values in the snapshot. You can move around within the data buffer by specifying an offset. You can indicate the absolute location of the data, or a location relative to the current position of the cursor within the buffer. You can also select a range of data, by specifying the starting and end offsets. For details on the dialog box options, see ["Go To Offset Dialog Box" on page 829](#).

Buffer Data Editing

You can perform all of the standard edit operations on buffer data: copy, paste, and delete. To perform edit operations on buffer data, right-click the required step in the Editor and select **Show Arguments**. You can then perform the edit operation in the **Text View** tab of the dialog box that opens. You cannot perform any edit operations in the Binary View tab.

Note: You perform edit operations on buffer data only, not on Snapshot data - which is read-only.

When you copy data from a buffer, VuGen allows you to copy the data either in hexadecimal format or in decimal format. When you insert data into a buffer, VuGen allows you to specify the format of the data—single byte, 2-byte, or 4-byte.

Record a Windows Sockets Script

This task describes optional steps that you can perform when recording using Windows Sockets.

For details on starting a recording session, see ["Record a Vuser Script" on page 135](#).

Before you start recording:

1. Open the recording options.

After creating a WinSock script, select **Record > Recording Options** and click the **WinSock** node.

2. Select a translation table.

In the **EBCDIC** section, select a translation table. If your data is in ASCII format, select the **None** option. Otherwise VuGen will convert the ASCII data. For details, see ["Translation Tables" on page 820](#).

3. Exclude any non-relevant sockets.

In the **Exclude Settings** section, add any non-relevant sockets to the list. You should exclude hosts and ports that do not influence the server load under test, similar to the local host and the DNS port (53), which are excluded by default.

To exclude the entries from the recording, but include them in the log, clear the **Do not include excluded sockets in log** option.

For user interface details, see the ["WinSock Recording Options" on page 203](#).

4. Set a think time threshold.

Indicate a think time threshold. If VuGen detects a pause in action less than the threshold time, it will not generate a **Think Time** step/ **lr_think_time** function. For details, see the ["WinSock Recording Options" on page 203](#).

After recording and saving the script:

1. Parameterize the script.

Replace recorded values with parameters using the shortcut menu. For more information, see ["Parameterization" on page 342](#).

2. Regenerate the script.

If you need to regenerate the script, for example if you want to include an excluded host:port, or if the translation was not correct:

- Select **Record > Regenerate Script**.
- Click the **Options** link.
- Under **General**, select **Protocols**, and then under **Active Protocols**, ensure that the **Windows Sockets** check box is selected.
- Under **Sockets**, select **Winsock**, and then modify the settings.

Note: Options for script regeneration are available for multi-protocol scripts only.

View and Modify Windows Sockets Buffers

The following steps describe how to view, modify, and navigate through WinSock buffer data.

Modifying buffer data

You can modify buffer data in the Show Arguments dialog box for specific **lrs** steps in a Vuser script. You can use the Show Arguments dialog box to modify buffer data for the following steps:

- lrs_length_receive
- lrs_length_send
- lrs_receive
- lrs_receive_ex
- lrs_send

For further details on these steps, see the Function Reference (**Help > Function Reference**).

To display the Show Arguments dialog box for any of the above steps, right-click a step in the Editor and select **Show Arguments**. A dialog box opens and displays the buffer data in the **Buffer Content** section of the dialog box.

For further details about editing buffer data in Windows Sockets steps, see ["Buffer Data Editing" on page 824](#).

Note: You cannot modify any data in the Snapshot pane

View and modify the data in the data.ws file

In the Solution Explorer, double-click the **data.ws** file. The contents of the data.ws file appear in the VuGen Editor. Modify the data directly in the Editor. For details, see ["Windows Sockets Data" on page 821](#).

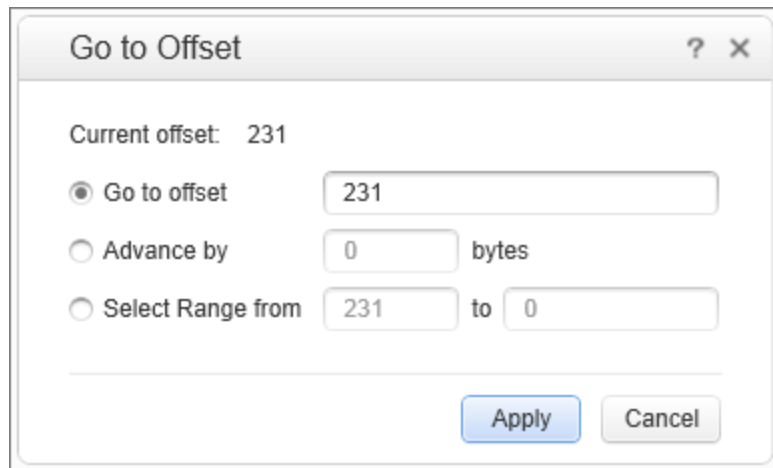
Note: Although it is possible to modify data.ws files, it is recommended that you do not modify these files.

View the data in the Snapshot pane

1. Ensure that the Snapshot pane is displayed.
2. Open the Vuser script in the Editor and select a step, or double-click an entry in the Step Navigator. The associated snapshot is displayed in the Snapshot pane. You cannot edit the snapshot data.

Navigate within the snapshot data

To navigate within the buffer data, display in the Snapshot pane, and then click **Go to**. The Go to Offset dialog box opens.



- To go to a specific offset within the buffer (absolute), select **Go to offset** and specify an offset value. Click **Apply**.
- To jump to a location relative to the cursor, click **Advance by** and specify the number of bytes to advance. To advance ahead, enter a positive value. To move backwards within the buffer, use a negative value. Click **Apply**.
- To select a range of data within the buffer, click **Select range from** and specify the beginning and end offsets. Click **Apply**.

Insert data into a buffer

You can insert a numerical value into a data buffer. You can insert the data as a single, double-byte, or 4-byte value. The following steps describe how to insert a number into a data buffer.

1. Copy the numerical data to be inserted to the clipboard.
2. Right-click a step in the Editor and select **Show Arguments**.
3. In the dialog box that opens, ensure that the **Text View** tab is displayed.
4. Under **Buffer Content**, right-click at the location in the buffer where you want to insert the data, and then select **Advanced > Paste Byte** or **Advanced > Paste Short (2 bytes)** or **Advanced > Paste Long (4 bytes)**.
5. Click **OK**. VuGen inserts the data into the buffer.

Copy and paste blocks of data

You can modify the buffer data as characters, decimal numbers, or hexadecimal numbers. For details, see ["Buffer Data Editing" on page 824](#).

Note: You can edit buffer data only when you view the step arguments. You cannot edit buffer data in the Snapshot pane.

1. Right-click a step in the Editor and select **Show Arguments**. A dialog box opens and displays the buffer data in the **Buffer Content** section of the dialog box.

2. To copy buffer data:
 - As characters, select one or more bytes and press Ctrl+c.
 - As a decimal number, select one or more bytes, right-click in the selection and select **Advanced > Copy As Decimal Number**.
 - As a hexadecimal number, select one or more bytes, right-click in the selection and select **Advanced > Copy As Hexadecimal Number**.
3. To paste the data:
 - As characters, press Ctrl+V.
 - As a single byte (assuming the size of the data on the clipboard is a single byte), right-click at the desired location in the buffer and click **Advanced > Paste Byte**.
 - In short format (2-byte), right-click at the desired location in the buffer and click **Advanced > Paste Short (2 bytes)**.
 - In long format (4-byte), right-click at the desired location in the buffer and click **Advanced > Paste Long (4 bytes)**.
4. To delete data, select the data in the Text view, right-click inside the selection and select **Delete**.

Data Buffers

When you use VuGen to create a Windows Sockets Vuser script, VuGen creates the **data.ws** data file. This file contains the data that is transmitted, and is expected to be received, during the replay sessions. You can right-click any step in the Editor and then select **Show Arguments** to show the buffer content that is stored in **data.ws** for the selected step. Using the **Text View** tab of the dialog box that opens, you can edit the data that is stored for any data buffer.

The **data.ws** data file has the following format:

- File header
- A list of buffers and their contents

The file header includes an internal version number of the data file format. The current version is 2. If you try to access data from a data file with format version 1, VuGen issues an error.

```
;WSRData 2 1
```

An identifier precedes each record, indicating whether the data was received or sent, followed by the buffer descriptor, and the number of bytes received (for **lrs_receive** only). The buffer descriptor contains a number identifying the buffer.

For example,

```
recv buf5 25
```

indicates that the buffer contains data that was received. The record number is 5, indicating that this receive operation was the sixth data transfer (the index is zero based), and twenty-five bytes of data were received.

If your data is in ASCII format, the descriptor is followed by the actual ASCII data that was transferred by the sockets.

If your data is in EBCDIC format, it must be translated through a look-up table. For information on setting the translation table, see the ["WinSock Recording Options" on page 203](#). The EBCDIC whose ASCII value (after translation) is printable, is displayed as an ASCII character. If the ASCII value corresponds to a non-printable character, then VuGen displays the original EBCDIC value.

```
recv buf6 39
"\xff\xfb\x01\xff\xfb\x03\xff\xfd\x01"
"\r\n"
"SunOS UNIX (sunny)\r\n"
```

The following segment shows the header, descriptors, and data in a typical data file:



Example:

```
;WSRData 2 1
send buf0 48
"\xff\xfd\x01\xff\xfd\x03\xff\xfb\x03\xff\xfb\x18"
recv buf1 15
"\xff\xfd\x18\xff\xfd\x1f\xff\xfd"
"#
"\xff\xfd"
""
"\xff\xfd"
"$"
send buf2 24
"\xff\xfb\x18"
```

Go To Offset Dialog Box

This dialog box allows you to go to a specific location within the recorded data.

To access	On the Snapshot pane toolbar, click Go to .
Relevant tasks	"View and Modify Windows Sockets Buffers" on page 826

User interface elements are described below:

UI Element	Description
Current offset	The current offset of the cursor (read only).
Go to offset	Goes to a specific, absolute offset within the data.

Advance by...bytes	Jumps to a location relative to the cursor, by a number of bytes. Positive values indicate a forward direction. Negative values indicate a reverse direction.
Select range from...to...	Selects a range of data within the buffer.
Apply	Moves the cursor to the specified offset.

Customize Your Scripts

This section provides you with tools to customize your scripts, such as how to manually program scripts and how to replay them on Linux machines.

What do you want to do?

- ["Create a PCAP File" below](#)
- ["Manually Program a Script using the VuGen Editor" on page 833](#)
- ["Create Scripts in External IDEs" on page 839](#)
- ["Use DLLs and Customize VuGen" on page 844](#)
- ["Create and Run Scripts in Linux" on page 848](#)
- ["Program with the XML API" on page 851](#)
- ["Share Software Content Resources" on page 868](#)



See also:

- ["Troubleshooting and Limitations for VuGen" on page 868](#)
- ["Non-English Language Support" on page 861](#)
- ["Additional Components" on page 869](#)

Create a PCAP File

Pcap (Packet Capture) files consist of network packet data, created by capturing live network activity through capture tools such as Wireshark. The generated .pcap file can be used for packet sniffing and analyzing network activity. VuGen can parse **.pcap** files and convert them to a Vuser script.

The primary uses for the .pcap file are:

- **Web Services scripts.** For details, see ["Create a Script by Analyzing Traffic \(Web Services\)" on page 737](#).
- **Mobile Applications scripts.** For details, see ["Create a Vuser Script by Analyzing a Captured Traffic File" on page 685](#).

This task describes how to create a **.pcap** capture file of network or application traffic to use in the preparation of a Vuser script.

Create a capture file

Windows

Create a capture file containing a log of all TCP traffic over the network on a Windows platform. Use a downloadable capture tool such as *Wireshark*. Make sure to save the Wireshark capture file in the *tcpdump* format, as this is the format supported by VuGen.

Note: For PCAP analysis, VuGen supports Wireshark versions Wireshark 2.2.x (up to and including version 2.2.7).

Android

Create a capture file using *tPacketCapture* on Android devices, or a similar application.

Linux Redhat

1. Copy the LinuxRH3 folder from LoadRunner installation disk **DVD\Additional Components\mobileRemoteAgent**, to the Linux machine.
2. Run the following commands in the shell to give the two files executable permissions on Linux.
 - a. `chmod +x mongoose`
 - b. `chmod +x cgi-bin/mobileCGI.cgi`
3. Perform a **Change directory** to the **cgi-bin/** folder and run the following commands:
 - a. `export SCRIPT_FILENAME=<Full path of mobileCGI.cgi>`, where `<Full path of mobileCGI.cgi>` is the path to the mobileCGI.cgi file.
 - b. `export QUERY_STRING=STARTRECORD=0`, where 0 is the 0-based sequence number of the network interface in VuGen's Start Recording dialog box.
 - c. `./mobileCGI.cgi` This generates a **currentPCAP.pcap** file which contains all traffic captured over the specified Ethernet interface.

Note: Step c. is primarily for troubleshooting, to make sure that your Linux environment was configured correctly. Once verified, you do not need to run this command again in future runs.

Tips for creating .pcap files

- To generate a smaller, more manageable script, try to capture the network traffic only for the time that you perform actions in your application.
- When using external tools, make sure that all packet data is being captured and none of it is being truncated.
- For command line capture utilities, make sure to provide all of the required arguments.

Troubleshooting missing packets

Issue: Your script is missing steps you recorded into a capture file.

You encounter the following warning in the **Output Pane> Code generation** tab:

Warning: One or more responses are missing or have missing packets. Therefore, a step

may appear to be missing in the script.
This issue can be caused if the recording was stopped before all the responses were received.
If the script is generated from a .pcap file, check if the file has missing packets.

This error may be caused by unnecessary network activity on the recorded machine, which can cause the capturing application to drop packets.

Steps to Resolve: Ensure that the capturing machine has no unnecessary network traffic in the background.

Workaround for Mobile Applications - HTTP/HTML scripts: You can circumvent this issue using the Recording options. Select **Recording Options > HTTP Properties > Advanced > Generate steps with missing responses** to generate steps for HTTP requests that are missing server responses.

Manually Program a Script using the VuGen Editor

Manually Programming Scripts - Overview

VuGen allows you to program your own functions into the script, instead of recording an actual session. You can use the Vuser API or standard programming functions. Vuser API functions allow you to gather information about Vusers. For example, you can use Vuser functions to measure server performance, control server load, add debugging code, or retrieve runtime information about the Vusers participating in the test or monitoring.

This chapter describes how to program a Vuser script from within the VuGen editor, incorporating your application's libraries or classes.

You can also develop a Vuser script through programming within the Visual C and Visual Basic environments. In these environments, you develop your Vuser script within your development application, while importing the Vuser API function libraries. For more information, see ["Create Scripts in External IDEs" on page 839](#).

To create a customized script, you first create a skeleton script. The skeleton script contains the three primary sections of a script: **init**, **actions**, and **end**. These sections are empty and you manually insert functions into them.

You can create empty scripts for the C and Java programming languages.



Tip: Make sure to define all variables at the beginning of the action, before all other API functions. This will prevent compilation errors.

Programming Vuser Actions

The Vuser script files, *test.c*, *test.usr*, and *test.cfg*, can be customized for your Vuser.

You program the actual Vuser actions into the *test.c* file. This file has the required structure for a programmed Vuser script. The Vuser script contains three sections: *vuser_init*, *Actions*, and *vuser_end*.

Note that the template defines extern C for users of C++. This definition is required for all C++ users, to make sure that none of the exported functions are modified inadvertently.



Example:

```
#include "lr.h"
#if defined(__cplusplus) || defined(cplusplus) extern "C"
{
#endif
int LR_FUNC vuser_init(LR_PARAM p)
{
    lr_message("vuser_init done\n");

    return 0;
}

int Actions(LR_PARAM p)
{
    lr_message("Actions done\n");

    return 0;
}

int vuser_end(LR_PARAM p)
{
    lr_message("vuser_end done\n");

    return 0;
}
#if defined(__cplusplus) || defined(cplusplus)}
#endif
```

You program Vuser actions directly into the empty script, before the **lr_message** function of each section.

The *vuser_init* section is executed first, during initialization. In this section, include the connection information and the logon procedure. The *vuser_init* section is only performed once each time you run the script.

The *Actions* section is executed after the initialization. In this section, include the actual operations performed by the Vuser. You can set up the Vuser to repeat the *Actions* section (in the *test.cfg* file).

The *vuser_end* section is executed last, after the all of the Vuser's actions. In this section, include the clean-up and logoff procedures. The *vuser_end* section is only performed once each time you run the script.



Note: Load Generators control Vusers by sending SIGHUP, SIGUSR1, and SIGUSR2 Linux

signals. Do not use these signals in your Vuser scripts.

Create a Template

VuGen includes a utility that copies a template into your working folder. The utility is called `mkdbtest`, and is located in `$M_LROOT/bin`. You run the utility by typing:

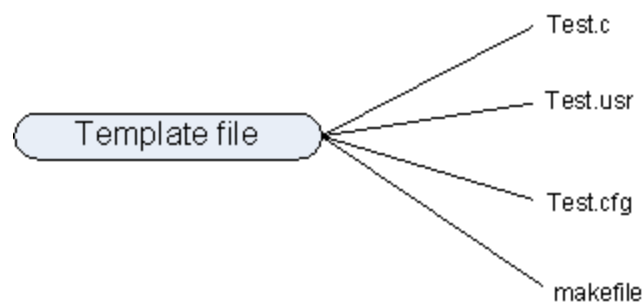
```
mkdbtest name
```

When you run `mkdbtest`, it creates a folder called `name`, which contains the template file, `name.c`. For example, if you type:

```
mkdbtest test1
```

mkdbtest creates a folder called *test1*, which contains the template script, *test1.c*.

When you run the *mkdbtest* utility, a folder is created containing four files *test.c*, *test.usr*, *test.cfg* and *Makefile*, where *test* is the test name you specified for *mkdbtest*.



Define Transaction and Insert Rendezvous Points Manually

When programming a Vuser script without VuGen, you must manually configure the Vuser file in order to enable transactions and rendezvous. The configuration settings are listed in the `test.usr` file.

```
[General]
Type=any
DefaultCfg=Test.cfg
BinVuser=libtest.libsuffix
RunType=Binary
[Actions]
vuser_init=
Actions=
vuser_end=
[Transactions]
transaction1=
[Rendezvous]
Meeting=
```

Each transaction and rendezvous must be defined in the *usr* file. Add the transaction name to the Transactions section (followed by an "="). Add each rendezvous name to the Rendezvous section (followed by an "="). If the sections are not present, add them to the *usr* file as shown above.

C Vuser Scripts

In a C Vuser script, you can use any C code that conforms to the standard ANSI conventions. To create an empty C Vuser script, select **C Vuser** in the Create a New Script dialog box. VuGen creates an empty C Vuser script:

```
Action1()  
{  
    return 0;  
}
```

You can use C Vuser functions in all of Vuser script types that use C functions.

See the Function Reference (**Help > Function Reference**) for a C reference that includes syntax and examples of commonly used C functions.

Guidelines for Using C Functions

All standard ANSI-C conventions apply to C Vuser scripts, including control flow and syntax. You can add comments and conditional statements to the script just as you do in other C programs. You declare and define variables using ANSI C conventions.

The C interpreter that is used to run Vuser scripts accepts the standard ANSI C language. It does not support any Microsoft extensions to ANSI C.

Before you add any C functions to a Vuser script, note the following limitations:

- A Vuser script cannot pass the address of one of its functions as a callback to a library function.
- The **stdargs**, **longjmp**, and **alloca** functions are not supported in Vuser scripts.
- Vuser scripts do not support structure or union arguments or return types. Pointers to structures are supported.
- In Vuser scripts, string literals are read-only. Any attempt to write to a string literal generates an access violation.
- C Functions that do not return int, must be casted. For example,
`extern char * strtok();`

Calling libc Functions

In a Vuser script, you can call **libc** functions. However, since the interpreter that is used to run Vuser scripts does not support any Microsoft extensions to ANSI C, you cannot use Microsoft's include files. You can either write your own prototypes, or ask Customer Support to send you ANSI-compatible include files containing prototypes for **libc** functions.

Linking Mode

The C interpreter that is used to run Vuser scripts uses a "lazy" linking mode in the sense that a function need not be defined at the start of a run, as long as the function is defined before it is used. For example:

```
lr_load_dll("mydll.dll");  
    myfun(); /* defined in mydll.dll -- can be called directly,  
            immediately after mydll.dll is loaded. */
```

Java Vusers

In Java Vuser scripts, you can place any standard Java code. To create an empty Java Vuser script, select **Java Vuser** in the New Virtual User dialog box. VuGen creates an empty Java script:

```
import lrapi.lr;  
public class Actions  
{  
    public int init() {  
        return 0;  
    }  
    public int action() {  
        return 0;  
    }  
    public int end() {  
        return 0;  
    }  
}
```




Note: For Java type Vusers, you can only edit the **Actions** class. Within the Actions class, there are three methods: **init**, **action**, and **end**. Place initialization code in the **init** method, business processes in the **action** method, and cleanup code in the **end** method.

.NET Vusers

You can create an empty .NET Vuser script, in which to place .NET code. This script type lets you incorporate your existing .NET application into VuGen. To create an empty .NET Vuser script, select **.NET** in the Create a New Script dialog box.

In a .NET Vuser script the default language is C#. If your script is generated from a recorded session, VuGen enables you to change the script language from C# to VB.NET by selecting Visual Basic .NET Language from **Record > Recording Options > General > Script** and regenerating the script.



Tip: You can edit the script in Visual Studio by clicking the  button.

The following example shows the Action section of an empty .NET script:

```
namespace Script
{
    public partial class VuserClass
    {
        public int Action()
        {
            // Add your code here
            return 0;
        }
    }
}
```

Note: Enter the business process code in the **Action** method. Add initialization code to the **vuser_init** method, and cleanup code to the **vuser_end** method.

Troubleshooting and Limitations - Programming

Framework 4.5 for .NET scripts

Issue: Running a .NET script recorded in VuGen, fails if you run it in Visual Studio—cannot find associated DLL.

Steps to Resolve: If the script was recorded with a .NET 4.5 AUT, rebuild the script with Framework version 4.5 in Visual Studio.

Framework 3.5 for .NET scripts

Issue: .NET DLLs created in Visual Studio using Framework 3.5 may not run.

Steps to Resolve: Add the following to the **<app>.config file**: (if there is no such file, create one)

```
<configuration>
<startup>
    <supportedRuntime version="v4.0" />
</startup>
</configuration>
```

Missing references for scripts created in Visual Studio

Issue: A compilation error occurs (such as error CS0246 or warning MSB3644) when compiling scripts created in Visual Studio, possibly because the referenced assemblies were not added correctly to Script.csproj.

Steps to Resolve:

- If Visual Studio is installed:
 - a. Open the script solution file in Visual Studio.
 - b. In the Solution Explorer, click **References** and select **Add Reference** from the right-click menu.
 - c. Save the solution, close Visual Studio, and reopen the script in VuGen.
- If Visual Studio is not installed:
 - a. Open the script.csproj file in a text editor.
 - b. In the **ItemGroup** element, add an Include statement for your reference as follows:

```
<Reference Include="ICSharpCode.SharpZipLib">  
<HintPath>C:\xxxx\ICSharpCode.SharpZipLib.dll</HintPath>  
</Reference>
```

- c. Save the file, close the editor, and reopen the script in VuGen.

Create Scripts in External IDEs

Creating Vuser Scripts or Unit Tests in Visual Studio or Eclipse

The following sections describe how to develop a Vuser script or unit test through programming within the Visual Studio or Eclipse environments.

LoadRunner provides add-ins that allow you to develop scripts with supported versions of Visual Studio or Eclipse.

Note: For details on supported Eclipse versions, see the [System Requirements](#).


There are two types of add-ins:


- Visual Studio/Eclipse IDE add-in
- Visual Studio/Eclipse IDE add-in for Developers

The basic **IDE add-in** allows you to create a Vuser script within the Visual Studio or Eclipse environment. You program a standard Vuser script using standard C# functions or protocol-specific functions from the Function Reference (**Help > Function Reference**), within your native environment. You can then use the script in your testing environment, for example, in a LoadRunner scenario.

The **IDE add-in for Developers** allows you to create *and run* a unit test directly from within Visual Studio or Eclipse. The run mechanism is VuGen's **mdrv** process. This add-in adds a **Devops** menu to the Visual Studio or Eclipse interface. You can configure runtime settings directly from your development environment and run the test. The saved tests, NUnit (Visual Studio) or JUnit (Eclipse), can be referenced directly from your testing environment, for example, a LoadRunner scenario.

These add-ins are provided in the main DVD folder of the product, under the **Additional Components** folder. Be sure to select the correct Visual Studio add-in for your version of Visual Studio.

Once you install the basic **IDE add-in**, you can create a new VuGen script within Visual Studio. Alternatively, you can begin developing your script in VuGen. If, while developing a script in VuGen, you realize that you need to the capabilities of your native environment, the **Open in Visual Studio**  or

Open in Eclipse  button opens the script in the respective application. This requires you to have first installed the basic **Visual Studio IDE add-in**. (For Eclipse, VuGen automatically installs the add-in the first time you choose **Open in Eclipse**). For details, see ["Debugging .NET Vuser Scripts" on page 582](#) or ["Opening Java Vuser Scripts in Eclipse" on page 539](#).

When working in Visual Studio or Eclipse, the complete VuGen API is available from the Object browser. For information about each of the Vuser functions that you can use when programming your script, see the Function Reference (**Help > Function Reference**).

Create a Vuser Script in Visual Studio

LoadRunner's basic IDE add-ins for Visual Studio let you create a Vuser script in Visual Studio in VB, C++, or C#.

Although .NET-based and Java protocols support creating threads, we recommend that you do not use background threads in real load testing scenarios because:

- Threads can degrade tests scalability
- Threads can affect performance measurements.
- The utility functions' behavior is undetermined if called from any thread except the Vuser main thread, which runs the `vuser_init`, `Action` and `vuser_end` actions. This applies to all functions named `lr*`.

To create a Vuser script in Visual Studio:

1. Install the IDE add-in for your version of Microsoft Visual Studio from the download/DVD **Additional Components** folder. For example, **Additional Components\IDE Add-Ins\LRVS<version>IDEAddInSetup.exe**.
2. In Visual Studio, select the appropriate template from the Installed Templates **LoadRunner VB|C++|C# .NET Vuser**. Visual Studio creates a new project with one class and a template for a Vuser, and the script file, `<name>.usr`. The template contains three sections, **Initialize**, **Actions**, and **Terminate**.

The following example shows a Visual C# template:

```
public int Initialize()
{
    // TO DO: Add virtual user's initialization routines
    return lr.PASS;
}
```

```
public int Actions()
{
    // TO DO: Add virtual user's business process actions
    return lr.PASS;
}
public int Terminate()
{
    // TO DO: Add virtual user's termination routines
    return lr.PASS;
}
```

3. Add code to the template, in the TODO sections.
4. Open the Object Browser (**View** menu). Expand the LoadRunner node (for example **Interop.LoadRunner**) to see the LoadRunner elements. Add the desired elements to your script, such as transactions, rendezvous points, and messages.
5. Expand the Toolbar menu, **Vuser**, and enhance your script with runtime settings and parameters. For more information, see the runtime settings **General > Run Logic** or the "[Parameter List Dialog Box](#)" on page 373.
6. Use the Vuser menu to replay the script and test its functionality.
7. Select **Vuser > Create Load Scenario**, to create a LoadRunner scenario using this .usr file.
8. You can also build the LoadRunner project as a DLL file, which will be saved in the same folder as the project. You can reference this DLL directly from a LoadRunner scenario.

Create a Vuser Script in Eclipse


LoadRunner's basic IDE add-in for Eclipse, lets you create a Java Vuser script in Eclipse. You can begin in VuGen and then open Eclipse. Alternatively, you can work from start to finish in Eclipse.

Note: For details on supported Eclipse versions, see the [System Requirements](#).

Prerequisite

1. Make sure you have JDK 1.7 (JRE 7) on your machine. Go to java.com to check your version or download the required version. After you install it, open Eclipse and select **Window > Preferences**. Navigate to the **Java > Installed JREs** node. If **jre7** is not in the **Installed JREs** list, click **Add** and use the wizard to add its folder (for example c:\Program Files\Java\jre7). In the Installed JREs list, click the check box by **jre7** to instruct Eclipse to use this version. Close Eclipse.

Method 1: Create a script in VuGen, and develop it further in Eclipse

1. Open VuGen and create a new Java type script.
2. Record or add steps as you normally would.
3. Click the Open in Eclipse button  on the toolbar. If this is the first time you are opening Eclipse

from VuGen, a message box prompts you to enter the Eclipse location. VuGen automatically copies the required files in Eclipse's **dropins** folder. Eclipse opens with your current script in the **Package Explorer** pane.

Method 2: Create the script in Eclipse

1. Manually copy the **hp.lr.vugeneclipse42addin.jar** file from the DVD's **Additional Components\IDE Add-Ins\EclipseAddin** folder into the Eclipse **dropins** folder. Extract the files from the jar file.
2. Open Eclipse. Select **File > New > Project** and expand the **LoadRunner Script** node. You can create any of the Java protocol scripts: Java over HTTP, Java Record Replay, or Java Vuser.



Tip: To verify that the add-in installed correctly, open the Eclipse Installation Details dialog box (Help > About > Installation Details.)

Develop the script in Eclipse

1. Expand the script's node and select the **default package > Action.java** node. Code the script as you normally would in the Eclipse editor, in the appropriate sections.
2. In the script's node in the Package Explorer, expand **Referenced Libraries > classes > lrapi >** to access the desired LoadRunner elements, such as transactions, rendezvous points, and messages.
3. Expand the **Vuser** menu (this may require you to select the parent script name in the Package Explorer) and enhance your script with runtime settings and parameters. For more information, see the runtime setting **General > Run Logic** or "[Parameter List Dialog Box](#)" on page 373.
4. Save and run the project. Select **Vuser > Run Vuser** to test the script. Then select **Vuser > Create Load Scenario** to run it from the Controller. Note that you will not be able to open this script in VuGen once you edit it in Eclipse.

Develop a Unit Test Using Visual Studio (NUnit test)

The LoadRunner Add-in for Developers lets you create an NUnit test in Visual Studio for use with LoadRunner.

To create an NUnit test in Visual Studio:

1. Install the IDE for Dev add-in for your Microsoft Visual Studio version from the download or DVD **Additional Components\IDE Add-Ins Dev** folder. For example, **Additional Components\IDE Add-Ins Dev\LRVS<version>IDEAddInDevSetup.exe**.
2. In Visual Studio, open your unit test. This test should comply with the following guidelines:
 - It is a class library
 - There is a reference to the NUnit library, `nunit.framework.dll`, **using nunit.framework;**
 - At least one of the classes in the project should be a `TestFixture` (using the `[TestFixture]` annotation)
3. In the code, instantiate the LoadRunner API function. For example,

```
private LoadRunner.LrApi lr = new LoadRunner.LrApi();
```

4. Select **DevOps Vuser > Add LoadRunner API Reference** to add protocol-specific or general API functions to your test. Alternatively, select **Add LoadRunner API Reference** from the context menu. Add LoadRunner functionality, such as transactions, think time, messaging, and so forth.
5. Build the LoadRunner project as a DLL file, which will be saved in the same folder as the project.
6. Select **DevOps Vuser > Run Vuser** to run the test with the LoadRunner engine. In the Visual Studio Output window, select **Show output from: LoadRunner Information** to view the runtime data.
7. (Optional) Add the DLL as a unit test to an existing or new LoadRunner scenario.

Develop a Unit Test Using Eclipse (JUnit or Selenium test)

The LoadRunner Eclipse Add-in for Developers lets you create a JUnit test in supported versions of Eclipse.

To create a unit test in Eclipse:

1. Make sure you have JDK 1.7 (JRE 7) on your machine. Go to **java.com** to check your version or download the required version. After you install it, open Eclipse and select **Window > Preferences**. Navigate to the **Java > Installed JREs** node. If **jre7** is not in the **Installed JREs** list, click **Add** and use the wizard to add its folder (for example c:\Program Files\Java\jre7). In the Installed JREs list, click the check box by **jre7** to instruct Eclipse to use this version.
2. Run the Eclipse Dev add-in, **LRclipseIDEAddInDevSetup.exe**, from the download/DVD folder: **Additional Components\IDE Add-Ins Dev**. After installing the Eclipse add-in, rebuild the plugin cache by running the following command line string: **Eclipse.exe -clean**.
3. In Eclipse, open your Selenium or JUnit test.
4. Code the test as you normally would in Eclipse.
5. Build your java classes.
6. Select **DevOps Vuser > Add LoadRunner API Reference** to add the desired LoadRunner functions to your script as well as transactions, rendezvous points, and messages.
7. Expand the **DevOps Vuser** menu and enhance the test with runtime settings and parameters. For more information, see the runtime setting **General > Run Logic** or the "[Parameter List Dialog Box](#)" on page 373.
8. Select **DevOps Vuser > Run Vuser** to run the test from within Eclipse to verify its functionality.
9. Use the **DevOps Vuser** menu to launch the LoadRunner Controller, or add the test to a Controller scenario that is already open.
10. Add the class file at any time as a unit test, to a LoadRunner scenario.

Note: If you are running multiple instances of Eclipse and you want to use add-in for each instance, you must manually install the Eclipse Add-in for Developers for each instance. Locate the **hp.lr.continuousdelivery.eclipse42addin.jar** file in the <LR_install_dir>\bin folder, and copy it to the **dropins** folder for each Eclipse instance.

Use DLLs and Customize VuGen

Calling Functions from External DLLs

You can call functions that are defined in external DLLs. By calling external functions from your script, you can reduce the memory footprint of your script and the overall runtime.

To call the external function, you load the DLL in which the function is defined.

You can load a DLL in one of the following ways:

- Locally (for one script) by using the **lr_load_dll** function. For task details, see ["Load a DLL Locally" below](#).
- Globally (for all scripts) by adding statements to the **vugen.dat** file. For task details, see ["Load a DLL Globally" on the next page](#).

Load a DLL Locally

This task describes how to use the **lr_load_dll** function to load a DLL into your Vuser script. Once the DLL is loaded, you can call any function defined within the DLL without having to declare it in your script.

To load a DLL locally:

1. In a C Vuser script, add an **lr_load_dll** function to load the DLL at the beginning of your script. Place the statement at the beginning of the *vuser_init* section. **lr_load_dll** replaces the **ci_load_dll** function.

Use the following syntax:

```
lr_load_dll( library_name);
```

Note that for Linux platforms, DLLs are known as shared libraries. The extension of the libraries is platform dependent.

2. Call the function defined in the DLL in the appropriate place within your script.

In the following example, the **insert_vals** function, defined in **orac1.dll**, is called, after the creation of the **Test_1** table.



Example: int LR_FUNC Actions(LR_PARAM p) { lr_load_dll("orac1.dll");

```
lrd_stmt(Csr1, "create table Test_1 (name char(15), id integer)\n", -1,  
1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0); lrd_exec(Csr1, 0, 0, 0, 0, 0);
```

```
/* Call the insert_vals function to insert values into the table. */
```



insert_vals0;

```
lrd_stmt(Csr1, "select * from Test_1\n", -1, 1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0); lrd_bind_col  
(Csr1, 1, =;NAME_D11, 0, 0); lrd_bind_col(Csr1, 2, =;ID_D12, 0, 0); lrd_exec(Csr1, 0, 0, 0, 0,  
0); lrd_fetch(Csr1, -4, 15, 0, PrintRow14, 0); ...
```



Note: You can specify a full path for the DLL. If you do not specify a path, **lr_load_library** searches for the DLL using the standard sequence used by the C++ function, LoadLibrary on Windows platforms. On Linux platforms you can set the **LD_LIBRARY_PATH** environment variable (or the platform equivalent). The **lr_load_dll** function uses the same search rules as **dlopen**. For more information, see the Unix help 'man' page for **dlopen**.

Load a DLL Globally

This task describes how to load a DLL globally, to make its functions available to all your Vuser scripts. Once the DLL is loaded, you can call any function defined within the DLL, without having to declare it in your script.

To globally load DLLs:

1. Add a list of the DLLs you want to load to the appropriate section of the *mdrv.dat* file, located in your application's *dat* folder.

Use the following syntax:

```
PLATFORM_DLLS=my_dll1.dll, my_dll2.dll, ...
```

replacing the word *PLATFORM* with your specific platform. For a list of platforms, see the beginning section of the *mdrv.dat* file.

For example, to load DLLs for Winsock Vusers on an NT platform, include the following section in the *mdrv.dat* file:



Example: [WinSock]


```
ExtPriorityType=protocol
```

```
WINNT_EXT_LIBS=wsrun32.dll
```

```
WIN95_EXT_LIBS=wsrun32.dll
```

```
LINUX_EXT_LIBS=liblrs.so
```

```
SOLARIS_EXT_LIBS=liblrs.so
```



```
HPUX_EXT_LIBS=liblrs.sl

AIX_EXT_LIBS=liblrs.so

LibCfgFunc=winsock_exten_conf

UtilityExt=lrapi ExtMessageQueue=0

ExtCmdLineOverwrite=-WinInet No

ExtCmdLineConc=-UsingWinInet No

WINNT_DLLS=user_dll1 .dll, user_dll2 .dll, ...
```

2. Call the function defined in the DLL in the appropriate place within your script.

VuGen File and Library Locations

The VuGen .dat files contain the location information of the script's files, as well as the library files for specific protocols.

There are two .dat files, residing in the **M_LROOT\dat** folder used by VuGen: **mdrv.dat** and **vugen.dat**.

mdrv.dat

The mdrv.dat file contains a separate section for each protocol defining the location of the library files and driver executables.

For information about how to add a custom protocol, see ["Protocol SDK" on page 875](#).

vugen.dat

The vugen.dat file contains general information about VuGen, used by VuGen and the Controller.

```
[Templates]
RelativeDirectory=template
```

The Templates section indicates where the templates are for the VuGen protocols. The default entry indicates that they are in the relative *template* folder. Each protocol has a subfolder under *template*, which contains the template files for that protocol.

The next section is the **GlobalFiles** section.

```
[GlobalFiles]
main.c=main.c
@@TestName@@.usr=test usr
default.cfg=test.cfg
default.usp=test.usp
```

The GlobalFiles section contains a list of files that VuGen copies to the test folder whenever you create a new test. For example, if you have a test called "user1", then VuGen will copy main.c, user1.usr and user1.cfg to the test folder.

The ActionFiles section contains the name of the file containing the Actions to be performed by the Vuser and upon which to perform iterations.

```
[ActionFiles]
@@actionFile@@=action.c
```

In addition to the settings shown above, *vugen.dat* contains settings that indicate the operating system and other compilation related settings.

Storing Runtime Settings in External Files

Vuser behavior refers to the items that you can set in the runtime settings, such as wait times, pacing times, looping iterations, and logging.

Since VuGen creates the Vuser script and the Vuser behavior as two independent sources, you can configure user behavior without directly referencing the Vuser script. This feature lets you make configuration changes to a Vuser and store several profiles for the same Vuser script.

VuGen stores the behavior settings in the default **Vuser.cfg** file. You can save several versions of this file for different user behavior and then run the Vuser script referencing the relevant *.cfg* file.

By default, you cannot control the behavior file from VuGen. VuGen automatically uses the *.cfg* file with the same name as the script.

To call a specific configuration file, run the Vuser from the command line and add the following string:

```
-cfg c:\tmp\<MyCustomConfigFile>.cfg
```

For information on command line parameters, see ["Command Line Parameters" below](#).

Note: The Linux utility, *run_db_vuser*, does not support this option.

Command Line Parameters

The Vusers can accept command line parameters when invoked.

For a complete list of the command line options, type **mdrv** at a command prompt from the installation's **bin** folder, without any arguments.

To send command line parameters to a Vuser from within VuGen, add the attributes and their values in the runtime settings. For details, see the **General > Additional Attributes** runtime settings view.

To control a Vuser from the Windows command line, type **mdrv** at a command prompt from the installation's **bin** folder, with the desired commands. You can also add custom user parameters, after all the other driver parameters. For example:

```
mdrv.exe -usr c:\tmp\Vuser\Vuser.usr -out c:\tmp\vuser command_line_params
```

There are several Vuser API functions available to reference them (such as **lr_get_attrib_double**, and so forth). For details, see the [Function Reference \(Help > Function Reference\)](#).

Note: The Linux utility, *run_db_vuser*, does not support some of the standard Windows command line options. For details, see ["Run a Vuser Script from a Linux Command Line" on the next page](#).

Create and Run Scripts in Linux

Creating and Running Scripts in Linux - Overview

You can use VuGen on a Linux environment in the following ways:

- You can use VuGen to create Vuser scripts that run on Linux platforms. You record your application in a Windows environment and run it in Linux—recording is not supported on Linux.

Note: VuGen provides a tool to check the compatibility of your script to run on Linux-based load generators. For details, see ["Check Linux Compatibility" on page 278](#).

- Users working in Linux-only environments can program Vuser scripts. Scripts can be programmed in C or C++ and they must be compiled into a dynamic library.

To create a script through programming, you can use a Vuser template as a basis for a larger Vuser scripts. The template provides:

- Correct program structure
- Vuser API calls
- Source code and makefiles for creating a dynamic library

Compile Scripts Manually on Linux

After you modify the template, you compile it with the appropriate *Makefile* in the script's folder. The compiler creates a dynamic library called **libtest.so**.

You can modify the *Makefile* and assign additional compiler flags and libraries by modifying the appropriate sections.

If you are working with a general template, you must include your application's libraries and header files. For example, if your application uses a library called *testlib*, include it in the LIBS section.

```
LIBS          = \  
-testlib \  
-
```

```
-lrun50 \  
-lm
```

After you modify the *Makefile*, type `make` from the command line in the working folder to create the dynamic library files for the Vuser script.

After you create a script, you check its functionality from the command line. Check that your script communicates with the server and performs all the required tasks. For details, see ["Run a Vuser Script from a Linux Command Line" below](#).

Run a Vuser Script from a Linux Command Line

When using VuGen to develop Linux-based Vusers, you must check that the recorded script runs on the Linux platform. This task describes how to perform this check and run a Vuser script from a Linux command.

1. Verify that the script replays in VuGen and is compatible for Linux

It is recommended to check the script in VuGen, before attempting to run it in Linux, because it is easier to edit and debug the script in VuGen.

- Replay the script in VuGen to verify that the script works in Windows. For details, see ["Replay a Vuser Script" on page 278](#).
- Run the Linux compatibility tool to check the compatibility of your script to run on Linux-based load generators. For details, see ["Check Linux Compatibility" on page 278](#).

2. Copy the script files to the Linux server

Transfer the script files to the Linux server.

3. Check the Vuser setup on the Linux machine by using `verify_generator`.

If you intend to run all of the Vusers on one host, type:

```
verify_generator
```

The `verify_generator` either returns **OK** when the setting is correct, or **Failed** and a suggestion on how to correct the setup.

For detailed information about the verify checks type:

```
verify_generator [-v]
```

The `verify` utility checks the local host for its communication parameters and its compatibility with all types of Vusers. It checks the following items in the Vuser environment:

- at least 128 file descriptors
- proper **.rhost** permissions: **-rw-r--r--**
- the host can be contacted using `rsh` to the host. If not, checks for the host name in **.rhosts**
- **M_LROOT** is defined

- **.cshrc** defines the correct **M_LROOT**
- **.cshrc** exists in the home directory
- the current user is the owner of the **.cshrc**
- a VuGen installation exists in **\$M_LROOT**
- the executables have executable permissions
- **PATH** contains **\$M_LROOT/bin**, and **/usr/bin**
- the **rstatd** daemon exists and is running

4. Run the script

Run the script in standalone mode from the Vuser script folder, using the **run_db_vuser** shell script:

```
run_db_vuser.sh <commands> script_name.usr
```

The **run_db_vuser** shell script has the following command line options:

Command	Description
--help	Display the available options. (This option must be preceded by two dashes.)
-cpp_only	Run cpp only (pre-processing) on the script.
-cci_only	Run cci only (pre-compiling) on the script to create a file with a .ci extension. You can run cci only after a successful cpp.
-driver driver_path	Use a specific external driver. Each database has its own driver located in the /bin folder.
-exec_only	Execute the Vuser .ci file. This option is available only when a valid .ci file exists.
-ci ci_file_ name	Execute a specific .ci file.
-out output_ path	Place the results in a specific folder.

By default, **run_db_vuser.sh** runs **cpp**, **cci**, and **execute** in verbose mode. It uses the driver in the VuGen installation\bin folder, and saves the results to an output file in the Vuser script folder. You must always specify a **.usr** file. If you are not in the script folder, specify the full path of the **.usr** file.

For example, the following command line executes a Vuser script called **test1**, and places the output file in a folder called **results1**. The results folder must be an existing folder—it will not be created automatically:

```
run_db_vuser.sh-out /u/joe/results1 test1.usr
```

Program with the XML API

Programming with the XML API Overview

VuGen's support for XML allows you to dynamically work with XML code and retrieve the values during test execution. Follow these steps in creating an effective XML script:

- Record a script in the desired protocol, usually Web, Web Services, or Wireless.
- Copy the XML structures into your script.
- Add XML functions from the LR API in order to retrieve dynamic data and the XML element values.

The LR API uses XPath, the XML Path language to manipulate the text in an XML document.

You can instruct VuGen to display the output values of XML elements in the Execution log window using the runtime settings. VuGen displays the line numbers, the number of matches, and the value. To allow the displaying of values, you need to enable parameter substitution. In the runtime settings, open the **General:Log** node, select **Extended log**, and select **Parameter Substitution**. For more information, see ["Runtime Settings Overview" on page 283](#).

All Vuser API XML functions return the number of matches found.

Using XML Functions

This section provides examples of how to work with data in an XML tree. Certain functions allow you to retrieve information, and others let you write information to an XML tree. The examples use the following XML tree containing the names and extensions of several employees in the Acme organization.

```
<acme_org>
  <accounting_dept>
    <employee type='PT'>
      <name>John Smith</name>
      <extension>2145</extension>
    </employee>
  </accounting_dept>
  <engineering_dept>
    <employee type='PT'>
      <name>Sue Jones</name>
      <extension>2375</extension>
    </employee>
  </engineering_dept>
</acme_org>
```

Reading Information from an XML Tree

The following functions read information from an XML tree:

lr_xml_extract	Extracts XML string fragments from an XML string.
lr_xml_find	Performs a query on an XML string.
lr_xml_get_values	Retrieves values of XML elements found by a query.

To retrieve a specific value through a query, you specify the tags of the parent and child nodes in a path format. For information about how to retrieve multiple values, see ["Multiple Query Matching " on page 854](#)

For example, to retrieve an employee name in the Accounting department, use the following string:

```
lr_xml_get_values("XML={XML_Input_Param}",
"ValueParam=OutputParam",
"Query=/acme_org/accounting_dept/employee/name",
LAST);
```

The Execution log window (with Extended logging enabled) shows the output of this function:

```
Output:
Action.c(20): "lr_xml_get_values" was successful, 1 match processed
Action.c(25): Query result = John Smith
```

Writing to an XML Tree

The following functions write values to an XML tree:

lr_xml_delete	Deletes fragments from an XML string.
lr_xml_insert	Inserts a new XML fragment into an XML string.
lr_xml_replace	Replaces fragments of an XML string.
lr_xml_set_values	Sets the values of XML elements found by a query.
lr_xml_transform	Applies Extensible Stylesheet Language (XSL) transformation to XML data.

The most common *writing* function is **lr_xml_set_values** which sets the values of specified elements in an XML string. The following example uses **lr_xml_set_values** to change the phone extensions of two *employee* elements in an XML string.

First, we save the XML string to a parameter called *XML_Input_Param*. We want two values to be matched and substituted, so we prepare two new parameters, *ExtensionParam_1* and *ExtensionParam_2*, and set their values to two new phone extensions, 1111 and 2222.

lr_xml_set_values contains the argument "ValueName=ExtensionParam", which picks up the values of ExtensionParam_1 and ExtensionParam_2. The current extensions of the two employees are substituted with the values of these parameters, 1111 and 2222. The value of OutputParam is then evaluated proving that the new phone extensions were in fact substituted.

```
Action() {
    int i, NumOfValues;
    char buf[64];

    lr_save_string(xml_input, "XML_Input_Param"); // Save input as parameter
    lr_save_string("1111", "ExtensionParam_1");
    lr_save_string("2222", "ExtensionParam_2");

    lr_xml_set_values("XML={XML_Input_Param}",
        "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
        "SelectAll=yes", "Query=//extension", LAST);

    NumOfValues= lr_xml_get_values("XML={NewXmlParam}",
        "ValueParam=OutputParam", "Query=//extension",
        "SelectAll=yes", LAST);

    for (i = 0; i < NumOfValues; i++) { /* Print the multiple values of MultiParam */

        sprintf(buf, "Retrieved value %d : {OutputParam_%d}", i+1, i+1);
        lr_output_message(lr_eval_string(buf));
    }

    return 0;
}
```

Output:

```
Action.c(40): Retrieved value 1: 1111
Action.c(40): Retrieved value 2: 2222
```

Specifying XML Function Parameters

Most XML API functions require that you specify the **XML element** and a **query**. You can also indicate if you want to retrieve all results or a single one.

Defining the XML Element

For defining the XML element to query, you can specify a literal string of the XML element, or a parameter that contains the XML. The following example shows the XML input string defined as a literal string,

```
"XML=<employee>JohnSmith</employee>"
```

Alternatively, the **XML** string can be a parameter containing the XML data. For example:

```
"XML={EmployeeNameParam}"
```

Querying an XML Tree

Suppose you want to find a value within an XML tag, for example, an employee's extension. You formulate a query for the desired value. The query indicates the location of the element and which element you want to retrieve or set. The path that you specify limits the scope of the search to a specific tag. You can also search for all elements of a specific type under all nodes below the root.

For a specific path, use `"Query=/full_xml_path_name/element_name"`

For the same element name under all nodes, use `"Query=//element_name"`

In the VuGen implementation of XML functions, the scope of a query is the entire XML tree. The tree information is sent to the Vuser API functions as the value of the *xml* argument.

Multiple Query Matching

When you perform a query on an XML element, by default VuGen returns only the first match. To retrieve multiple values from a query, you specify the `"SelectAll=yes"` attribute within your functions. VuGen adds a suffix of *_index* to indicate multiple parameters. For example, if you defined a parameter by the name *EmployeeName*, VuGen creates *EmployeeName_1*, *EmployeeName_2*, *EmployeeName_3*, and so on.

```
lr_xml_set_values("XML={XML_Input_Param}",  
"ResultParam=NewXmlParam", "ValueParam=ExtensionParam",  
"SelectAll=yes", "Query=//extension", LAST);
```

With functions that *write* to a parameter, the values written to the parameter can then be evaluated. For example, the following code retrieves and prints multiple matches of a query:

```
NumOfValues = lr_xml_get_values("Xml={XmlParam}", "Query=//name",  
"SelectAll=yes", "ValueParam=EmployeeName", LAST);
```

For functions that *read* from parameters, the values of the parameters must be pre-defined. The parameter must also use the convention *ParamName_IndexNumber*, for example *Param_1*, *Param_2*, *Param_3*, and so on. This collection of parameters is also known as a parameter set.

In the following example, `lr_xml_set_values` reads values from the parameter set and then uses those values in the XPath query. The parameter set that represents the employee extensions, is called *ExtensionParam*. It has two members: *ExtensionParam_1* and *ExtensionParam_2*. The **`lr_xml_set_values`** function queries the XML input string and sets the value of the first match to 1111 and the second match to 2222.

```
lr_save_string("1111", "ExtensionParam_1");  
lr_save_string("2222", "ExtensionParam_2");
```

```
lr_xml_set_values("XML={XML_Input_Param}",  
    "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",  
    "SelectAll=yes", "Query=//extension", LAST);
```

XML Attributes

VuGen contains support for attributes. You can use a simple expression to manipulate attributes of XML elements and nodes, just as you can manipulate the elements themselves. You can modify the desired attribute or only attributes with specific values.

In the following example, **lr_xml_delete** deletes the first cubicle element with the name attribute.



Example:

```
lr_xml_delete("Xml={ParamXml}",  
    "Query=//cubicle/@name",  
    "ResultParam=Result",  
    LAST  
    );
```

In the next example, **lr_xml_delete** deletes the first cubicle element with a name attribute that is equal to Paul.



Example:

```
lr_xml_delete("Xml={ParamXml}",  
    "Query=//cubicle/@name=\"Paul\"",  
    "ResultParam=Result",  
    LAST  
    );
```

Structuring XML Scripts

Initially, you create a new script in your preferred protocol. You can record a session in that protocol, or you may program the entire script without recording. Structure the Actions section of the script as follows:

- XML input declaration
- The Actions section

The XML input section contains the XML tree that you want to use as an input variable. You define the XML tree as a char type variable. For example:

```
char *xml_input=  
    "<acme_org>"  
    "<employee>"  
        " <name>John Smith</name>"  
        "<cubicle>227</cubicle>"
```

```
"<extension>2145</extension>"
"</employee>"
"<employee>"
  "<name>Sue Jones</name>"
  "<cubicle>227</cubicle>"
  "<extension>2375</extension>"
"</employee>"
"</acme_org>";
```

The Action section contains the evaluation of the variables and queries for the element values. In the following example, the XML input string is evaluated using **lr_save_string**. The input variable is queried for employee names and extensions.

```
Action() {

    /* Save the input as a parameter.*/
    lr_save_string(xml_input, "XML_Input_Param");
    /* Query 1 - Retrieve an employee name from the specified element.*/
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=/acme_org/employee/name", LAST);

    /* Query 2 - Retrieve an extension under any path below the root.*/
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=//extension", LAST);

    return 0;
}
```

Enhancing a Recorded Session with XML

You can prepare an XML script by recording a session and then manually adding the relevant XML and Vuser API functions.

The following example illustrates how a recorded session was enhanced with Vuser API functions. Note that the only function that was recorded was **web_submit_data**, which appears in bold.

The first section contains the XML input declaration of the variable SOAPTemplate, for a SOAP message:

```
#include "as_web.h"
// SOAP message
const char* pSoapTemplate=
    "<soap:Envelope xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">"
    "<soap:Body>"
    "<SendMail xmlns=\"urn:EmailIPortTypeInft-IEmailService\"/>"
```

```
"</soap:Body>"
"</soap:Envelope>";
```

The following section represents the actions of the user:



Example:

```
Action1()
{
// get response body
web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);
// fetch weather by HTTP GET
web_submit_data("GetWeather","Action=http://glkev.net.innerhost.com/glkev_ws/
WeatherFetcher.asmx/GetWeather",
"Method=GET",
"EncType=",
"RecContentType=text/xml",
"Referer=http://glkev.net.innerhost.com/glkev_
ws/WeatherFetcher.asmx?op=GetWeather",
"Snapshot=t2.inf",
"Mode=HTTP",
ITEMDATA,
"Name=zipCode", "Value=10010", ENDITEM,
LAST);

// Get City value
lr_xml_get_values("Xml={ParamXml}",
"Query=City",
"ValueParam=ParamCity",
LAST
);
lr_output_message(lr_eval_string("***** City = {ParamCity} *****"));

//Get State value
lr_xml_get_values("Xml={ParamXml}",
"Query=State",
"ValueParam=ParamState",
LAST
);
lr_output_message(lr_eval_string("***** State ={ParamState}*****"));

// Get several values at once by using template

lr_xml_get_values_ex("Xml={ParamXml}",

"Template="
"<Weather>"
"<Time>{ParamTime}</Time>"
```



```

        "<Temperature>{ParamTemp}</Temperature>"
        "<Humidity>{ParamHumid}</Humidity>"
        "<Conditions>{ParamCond}</Conditions>"
    "</Weather>",
    LAST
);

lr_output_message(lr_eval_string("***** Time = {ParamTime},
    Temperature = {ParamTemp}, "
    "Humidity = {ParamHumid},
    Conditions = {ParamCond} *****"));

//Generate readable forecast
    lr_save_string(lr_eval_string("\r\n\r\n*** Weather Forecast for {ParamCity},
        {ParamState} ***\r\n"
        "\tTime: {ParamTime}\r\n"
        "\tTemperature: {ParamTemp} deg. Fahrenheit\r\n"
        "\tHumidity: {ParamHumid}\r\n"
        "\t{ParamCond} conditions expected\r\n"
        "\r\n"),
        "ParamForecast"
    );
// Save soap template into parameter
    lr_save_string(pSoapTemplate, "ParamSoap");

// Insert request body into SOAP template
    lr_xml_insert("Xml={ParamSoap}",
        "ResultParam=ParamRequest",
        "Query=Body/SendMail",
        "position=child",
        "XmlFragment="
        "<FromAddress>john1@hpe.com</FromAddress>"
        "<ToAddress>softwaresupport@hpe.com</ToAddress>"
        "<ASubject>Weather Forecast</ASubject>"
        "<MsgBody/>",
        LAST);
    "<soap:Envelope"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" "<soap:Body>" "<SendMai
    1 xmlns="urn:EmailIPortTypeInft-
    IEmailService"/>" "<FromAddress>john1@hpe.com</FromAddress>"
        "<ToAddress>softwaresupport@hpe.com</ToAddress>"
        "<ASubject>Weather Forecast</ASubject>"
        "<MsgBody/>"
        "</SendMail>"
    "</soap:Body>" "</soap:Envelope>";

```



```
// Insert actual forecast text
lr_xml_set_values("Xml={ParamRequest}",
                 "ResultParam=ParamRequest",
                 "Query=Body/SendMail/MsgBody",
                 "ValueParam=ParamForecast",
                 LAST);

// Add header for SOAP
web_add_header("SOAPAction", "urn:EmailIPortTypeInft-IEmailService");

// Get response body
web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);

// Send forecast to recipient, using SOAP request
web_custom_request("web_custom_request",
                  "URL=http://webservices.matlus.com/scripts/emailwebservice.dll/soap
                  /IEmailservice",
                  "Method=POST",
                  "TargetFrame=",
                  "Resource=0",
                  "Referer=",
                  "Body={ParamRequest}",
                  LAST);

// Verify that mail was sent
lr_xml_find("Xml={ParamXml}",
           "Query=Body/SendMailResponse/return",
           "Value=0",
           LAST);

return 0;
}
```

Use Result Parameters

Some of the **lr_xml** functions return a result parameter, such as **ResultParam**. This parameter contains the resulting XML data after the function is executed. The result parameters will be available from the parameter list in the Select or Create Parameter dialog box.

For example, for **lr_xml_insert**, ResultParam contains the complete XML data resulting from the insertion of the new XML fragment

You can use the result parameters as input to other XML related functions such as Web Service calls. During replay, VuGen captures the value of the result parameter. In a later step, you can use that value as an input argument.

The functions that support result parameters are **lr_xml_insert**, **lr_xml_transform**, **lr_xml_replace**, **lr_xml_delete**, and **lr_xml_set_values**.

The following functions save values to a parameter other than the resultParam: **lr_xml_get_values** saves values to ValueParam and **lr_xml_extract** saves values to XMLFragmentParam. These values are also available for parameter substitution.

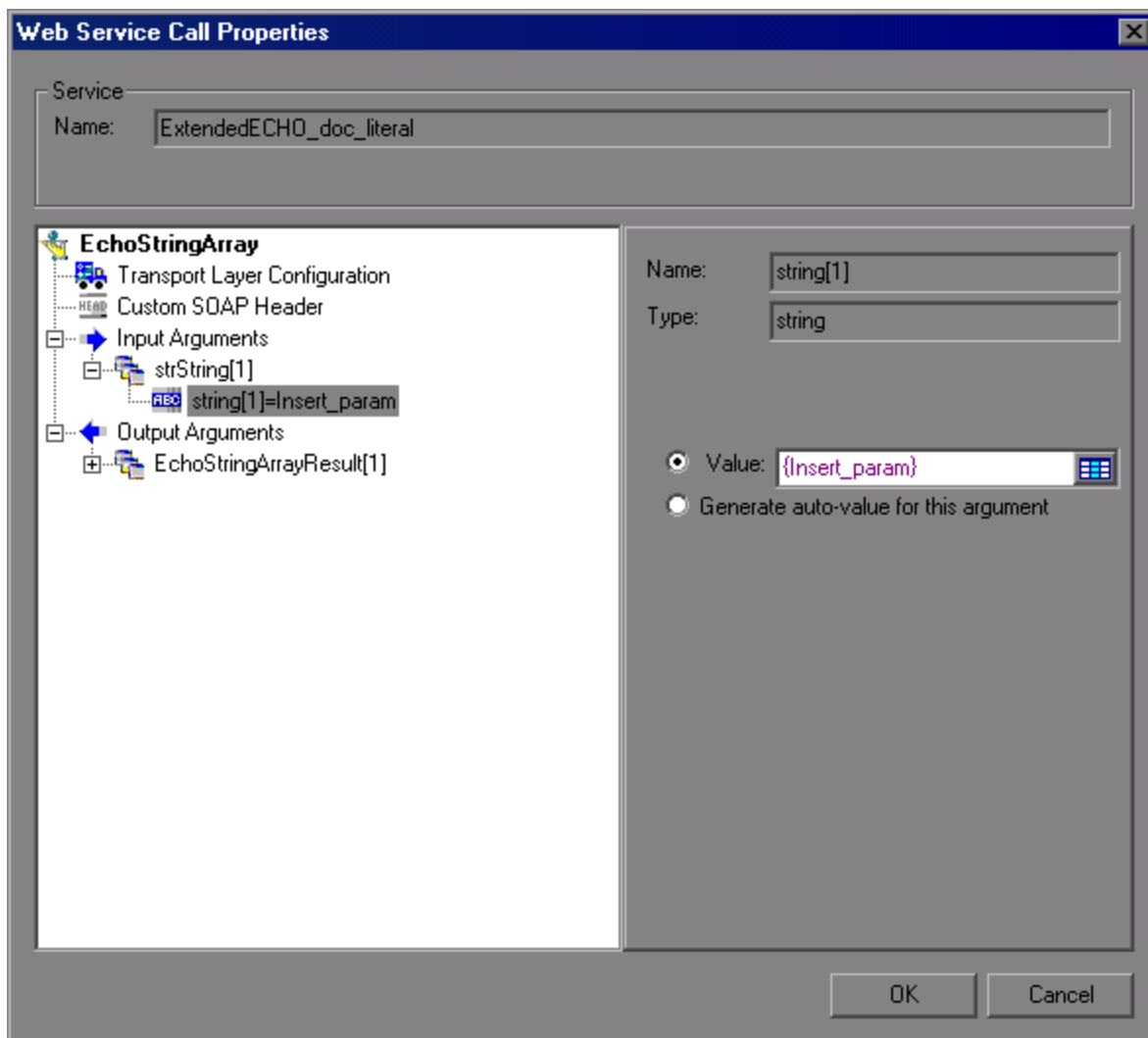
Use the Result Parameter as Input

1. In the **Step Navigator**, double-click on an XML step to view its Properties.
2. In the Result XML Parameter box, specify a name for the **Result XML parameter** (or ValueParam and XMLFragmentParam).

The screenshot shows the 'Insert XML' dialog box. It contains the following fields and controls:

- XML:** A text field containing 'Click to Edit' and an 'Edit...' button.
- XPath query:** A text field containing '/acme/org/' and an 'ABC' button.
- XML fragment:** A text field containing '<extension>245</e>' and an 'Edit...' button.
- XML fragment parameter:** An empty text field and an 'ABC' button.
- Result XML parameter:** A text field containing 'Insert_param' and an 'ABC' button.
- Position:** A dropdown menu with 'child' selected and an 'ABC' button.
- Select all:** A dropdown menu with 'no' selected and an 'ABC' button.
- NotFound:** A dropdown menu with 'error' selected and an 'ABC' button.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

3. Reference the parameter name as in input argument.



For more information, see ["New Web Service Call Dialog Box"](#) on page 740.

Non-English Language Support

Non-English Language Support Overview

VuGen supports multilingual environments, allowing you to use languages other than English on native language machines when creating and running scripts.

When working with languages other than English, the primary issue is ensuring that VuGen recognizes the encoding of the text during record and replay. The encoding applies to all texts used by the script. This includes texts in HTTP headers and HTML pages for Web Users, data in parameter files, and others.

If you need to use non-English symbols in paths to scripts, scenarios, results, or analysis sessions, make sure to select the appropriate locale in the your machine's **Region and Language** settings. Script names, however, must be in English.

Windows 2000 and higher lets you save text files with a specific encoding directly from Notepad: ANSI, Unicode, Unicode big endian, or UTF-8.

By default, VuGen works with the local machine encoding (ANSI). Some servers working with foreign languages, require you to work with UTF-8 encoding. To work against this server, you must indicate in the Advanced recording options, that your script requires UTF-8 encoding.

Page Request Header Language

Before running a Web script, you can set the page's request header to match your current language. In the Internet Protocol runtime settings, you set the language of the *Accept-Language* request header. This header provides the server with a list of all of the accepted languages.

To set this value, select **Replay > Runtime Settings > Internet Protocol > Preferences > Advanced > Options > Accept-Language request header** and select the desired language.

For user interface details, see "[Preferences View - Internet Protocol](#)" on page 291.

Convert Encoding Format of a String

You can manually convert a string from one encoding to another (UTF-8, Unicode, or locale machine encoding) using the **lr_convert_string_encoding** function with the following syntax:

```
lr_convert_string_encoding(char * sourceString, char * fromEncoding, char * toEncoding,  
char * paramName)
```

The function saves the result string (including its terminating NULL) in the third argument, *paramName*. It returns a 0 on success and -1 on failure.

The format for the **fromEncoding** and **toEncoding** arguments are:

LR_ENC_SYSTEM_LOCALE	NULL
LR_ENC_UTF8	"utf-8"
LR_ENC_UNICODE	"ucs-2"

In the following example, **lr_convert_string_encoding** converts "Hello world" from the system locale to Unicode.

```
Action()  
{  
    int rc = 0;
```

```
    unsigned long converted_buffer_size_unicode = 0;
    char          *converted_buffer_unicode = NULL;
    rc = lr_convert_string_encoding("Hello world", NULL, LR_ENC_UNICODE,
"stringInUnicode");
    if(rc < 0)
    {
        // error
    }
    return 0;
}
```

In the replay log, the output window shows the following information:

Output:

Starting action Action.

Action.c(7): Notify: Saving Parameter "stringInUnicode = H\x00e\x00l\x00l\x00o\x00
\x00w\x00o\x00r\x00l\x00d\x00\x00"

Ending action Action.

The result of the conversion is saved to the *paramName* argument.

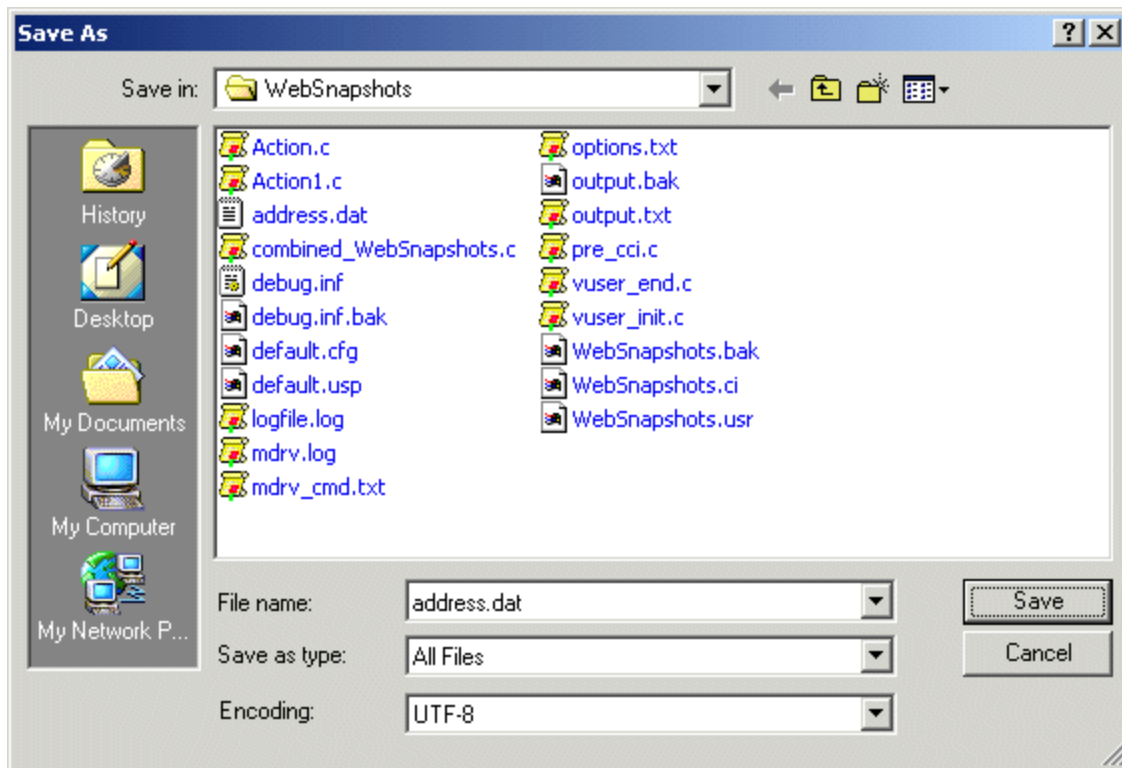
Convert Encoding Format of Parameter Files

The parameter file contains the data for parameters that were defined in the script. This file, stored in the script's folder, has a *.*dat* extension. When running a script, Vusers use the data to execute actions with varying values.

By default, VuGen saves the parameter file with your machine's encoding. When working with languages other than English, however, in cases where the server expects to receive the string in UTF-8, you may need to convert the parameter file to UTF-8. You can do this directly from Notepad, provided that you are working with Windows 2000 or higher.

Apply UTF-8 Encoding to a Parameter File

1. Select **Vuser > Parameter List** and view the parameter properties.
2. In the right pane, locate the parameter file in the **File path** box.
3. With the parameter table in view, click **Edit in Notepad**. Notepad opens with the parameter file in csv format.
4. In the **Save as type** box, select *All Files*.
In the **Encoding** box, select *UTF-8* type encoding.



5. Click **Save**. Notepad asks you to confirm the overwriting of the existing parameter file. Click **Yes**.
VuGen now recognizes the parameter file as UTF-8 text, although it still displays it in regular characters.

Record Web Pages with Foreign Languages

When working with Web or other Internet protocols, you can indicate the encoding of the Web page text for recording. The recorded site's language must match the operating system language. You cannot mix encodings in a single recording—for example, UTF-8 together with ISO-8859-1 or shift_jis.

Automatically Record Foreign Language Web Pages.

To be recognized as a non-English Web page, the page must indicate the charset in the HTTP header or in the HTML meta tag. Otherwise, VuGen will not detect the EUC-JP encoding and the Web site will not be recorded properly. To instruct VuGen to record non-English requests as **EUC-JP** or **UTF-8**, select **Record > Recording Options > HTTP Properties > Advanced > support charset** and select the appropriate option in the Recording Options dialog box, **HTTP Properties: Advanced** node. For user interface details, see ["HTTP Properties > Advanced Recording Options" on page 175](#).

By selecting the **EUC-JP** or **UTF-8** option in the Recording Options, you are forcing VuGen to record a Web page with the selected encoding, even when it uses different encoding. If, for example, a non-EUC encoded Web page is recorded as EUC-JP, the script will not replay properly.

Manually Record Foreign Language Web Pages

You can manually add full support for recording and replaying of HTML pages encoded in EUC-JP using the **web_sjis_to_euc_param** function. This also allows VuGen to display Japanese EUC-encoded characters correctly in Vuser scripts.

When you use **web_sjis_to_euc_param**, VuGen shows the value of the parameter in the Execution Log using EUC-JP encoding. For example, when you replay the **web_find** function, VuGen displays the encoded values. These include string values that were converted into EUC by the **web_sjis_to_euc_param** function, or parameter substitution when enabled in the **RuntimeSetting > Log > Extended Log**.

Troubleshooting and Limitations for Non-English Languages

This section describes troubleshooting and limitations when working with non-English languages.

Script / Scenario Names

- When recording COM, FTP, IMAP, SMTP, or POP3 protocols, the length of the script name is limited to 10 multi-byte characters (21 bytes).
- If you need to use non-English symbols in paths to scripts, scenarios, results, or analysis sessions, make sure to select the appropriate locale in the your machine's **Region and Language** settings. Script names, however, must be in English.
- The name and path of a scenario cannot contain multi-byte characters. . It is also recommended to use English characters for argument and parameter names.
- The GWT DFE extension does not support non-English characters in its classpath.

Browser Configuration

If, during recording, non-English characters in the script are displayed as escaped hexadecimal numbers (For example, the string " =;" becomes "%DC%26"), you can correct this by configuring your browser not to send URLs in UTF-8 encoding. In Internet Explorer, select **Tools > Internet Options** and click the **Advanced** tab. Clear the **Always Send URLs as a UTF-8** option in the Browsing section. For more information, use the **web_sjis_to_euc_param** function described in the Function Reference.

Jenkins Reports

After running a LoadRunner job in Jenkins, the links to several reports, such as the Performance Report and Transaction Summary, may not be translated.

Protocol Limitations

SMTP: If you work with the SMTP protocol through MS Outlook or Outlook Express, the Japanese text recorded in a Vuser script is not displayed correctly. However, the script records and replays correctly.

ContentCheck in Multilingual Environments

- This version supports ContentCheck rules in French, German, Spanish, and Italian. The correct language file should be installed according to the system locale.

- The suitable language file can also be copied from the installation disk:
 `..\runner\MS\setup\international\<lang>\dat\LrwiAedInstallation.xml` to the product's **dat** directory.

Language Packs

- **LoadRunner Language Pack.** While installing the language pack, a warning message may be displayed that the HPE LoadRunner Launcher Process is in use.
Workaround: Click **Continue** to resume the installation.
- **Match Windows locale.** Install the language pack that is appropriate for the Windows locale. For example, if you are installing the Russian language pack, the Windows locale must also be Russian.
- **Framework Language Pack.** The language pack of .NET Framework needs to be installed to show the localized strings.
- **License Utility warnings.** A License Utility dialog box may open when installing a language pack. It will not affect the installation. Close it, and continue installing the language pack.
- **Certificate warnings.** VuGen may issue warnings about certificates when recording a Web - HTTP/HTML script on Windows 2012 R2 and Windows 8.1. This only occurs when you install VuGen on a non-English operating system.
Workaround: Install the language pack for the language of the operating system. Do not delete the certificate after recording.
- **Recording Functions:** VuGen cannot record a Vuser script for certain protocols if the installation is on a Chinese operating system, and the installation path contains Chinese characters.
- **Tutorial scripts.** After the language pack installation in LoadRunner, all sessions and scripts in `\HP\LoadRunner\tutorial` will remain in English.
- **Menus and toolbars.** If you install a language pack after running VuGen for the first time, the menus and toolbars may not get translated.
Workaround: Close the application and delete the following folder from the registry: `HKEY_CURRENT_USER\Software\<Folder Name>`, where `<Folder Name>` is the drive on which you installed the product.
For example, if LoadRunner is installed on the C drive, the registry folder name would be: `HKEY_CURRENT_USER\Software\C`. Restart VuGen.

Non-Localized Installations on Foreign Language Operating Systems

- **Language support.** VuGen supports English and the native language of the machine's operating system. For example, if you are using Japanese Windows XP, you can work with VuGen in Japanese and in English.
- **Installation path.** The path in which installation files for VuGen are located, and the path in which LoadRunner is installed, can contain only English characters.
- **Diagnostics add-in.** To use the Diagnostics add-in with Controller on a computer with a non-English operating system the `Diagnostics_9.0_8.0_LR_Addin_QCCR1I52206` hotfix should be installed. For further assistance, contact HPE customer support.
- **.NET Framework 3.5 failure.** Installing VuGen on a localized machine may result in a failure in the

.NET Framework 3.5 installation process, and you will be asked to terminate the installation. This happens because the .NET 3.5 Framework installation attempts to download the Framework Language Pack but fails.

Workaround: Terminate the VuGen installation according to the Installation wizard's instructions and invoke the VuGen installation again.

- **Online Help.** The search functionality may not function as expected for strings that contain Chinese/Japanese characters (except Japanese full-width Katakana).
Workaround: Add a half-width space after each character in the search string.
- **Online Help.** For optimum performance of the online Help, install the latest JRE.
- **Japanese characters in Web - HTTP/HTML scripts.** If you set the advanced recording option to specify the encoding of an application, and the application uses different character encoding for different pages, then the recording log or script may display invalid Japanese characters. This does not cause any errors in the script replay.
- **Non-breaking spaces** in Web protocols for Far Eastern languages. A non-breaking space ('\xA0', etc.) cannot be represented in some Far Eastern locale character sets (in which it is considered a lead byte). Instead, non-breaking spaces are converted to regular spaces (' ', '\x20', etc.), both during script code generation and replay. This may cause replay problems, such as mismatches in length due to eliminating multiple regular spaces.
Workaround: Remove/add space(s) from/to the script so the comparison succeeds or specify regular expressions to avoid the issue.
- **Standalone installations.** The installation interface of the VuGen standalone is in English and not localized.
- **Flex AMF call properties.** Multibyte symbols in Flex AMF call properties will be corrupted in the script text view.
- **rdp_type** The **rdp_type** function does not support native language characters for both record and replay.
- **Word Completion.** Word completion does not work when Windows is configured to use the Ctrl+Space combination. This is common when using a Chinese keyboard.
Workaround: Select **Complete word** from the **Edit** menu. Advanced users can disable Ctrl+Space for Chinese keyboards, by setting the following registry keys:
 - [HKEY_CURRENT_USER\Control Panel\Input Method\Hot Keys\00000010]
"Key Modifiers"=hex:00,c0,00,00
"Target IME"=hex:00,00,00,00
"Virtual Key"=hex:ff,00,00,00
 - [HKEY_CURRENT_USER\Control Panel\Input Method\Hot Keys\00000070]
"Key Modifiers"=hex:00,c0,00,00
"Target IME"=hex:00,00,00,00
"Virtual Key"=hex:ff,00,00,00
- **ODBC and Oracle-2 Tier protocols.** When recording a script in VuGen using the ODBC or Oracle-2 Tier protocols, if you stop the recording while the AUT is still open, VuGen may crash.
Workaround: Close VuGen and open the file <installation folder>\dat\protocols\options\script\general.opt in a text editor.
Comment out the following line by adding a semicolon at the beginning of the line:

Option=DumpProcesses so it looks like this: ;Option=DumpProcesses

Share Software Content Resources

Share software content resources with other VuGen users using HPE AppDelivery Marketplace. You can download or contribute all types of content, including new protocols, documents, installation packages, and so on.

To download content from Marketplace:

1. Go to Marketplace > Performance Engineering:
 - In VuGen: Click the **Marketplace** button on the toolbar, or select **Help > AppDelivery Marketplace**
 - Access directly: <https://marketplace.saas.hpe.com/appdelivery/category/performance-engineering>.
2. Click a file to download, and click **Download**.

To contribute content to Marketplace:

1. **Prerequisite:** Join Marketplace as a developer: <https://marketplace.saas.hpe.com/appdelivery/join>
2. On your **Dashboard** page, click **ADD NEW ITEM**.
3. Give the item a name.
4. Complete the form and submit for review.

Troubleshooting and Limitations for VuGen

This section describes general troubleshooting and limitations for VuGen. For additional protocol-specific limitations, see the troubleshooting sections for each of the protocols.

Internet Explorer and Windows Server Machines

When using Internet Explorer on Windows server machines, the browser's enhanced security (ESC) blocks certain actions. This may prevent the automatic download of files that are necessary for your workflow.

Error Messages

For Media Player - MMScripts, if you specify a non-default bandwidth in the runtime settings, the Vuser may cause an error during replay.

Slow replay of JavaScript language scripts

If your JavaScript language script runs slow in VuGen, disable the debugging option: In the **Internet Protocol** runtime settings, open the **Preferences** view and locate the JavaScript section. Clear the **Enable JavaScript debugging mode** option. For details, see "[Preferences View - Internet Protocol](#)" on page 291.

Installing and Upgrading JVMs

If you install or upgrade a JVM while VuGen is open, you will need to restart VuGen before continuing to record or develop a script.

Workaround: Add an entry "**about:internet**" to the Trusted Sites in Internet Explorer.

McAfee Compatibility Issues

- When McAfee On-Access file scan is enabled, it may block/revert some of the file-writing operations that VuGen performs during code generation, particularly when asynchronous communications have been detected. As a result, the content of some of the script actions may be lost; these actions will be empty instead of containing the required generated Async code.

Workaround: Exclude LoadRunner files from the scan.

- When McAfee Host Intrusion Prevention (HIP) is enabled, a crash may occur while recording a Web-based protocol script, including Java over HTTP, especially if the current user account is a limited account (non-admin). In addition, you may experience browser malfunctions, even when working with a full-privileged user account.
- It is recommended that you close all anti-virus applications, such as McAfee or Aladdin's eSafe, before installing LoadRunner.
- McAfee's anti-virus application blocks port 443, which is the default port of the LoadRunner agent.

Workaround: Manually enable this port. To enable the port, open the McAfee Configuration dialog box. In the Firewall Policy tab, add a new rule to allow Port 443 - Action: Permit IP: TCP, Incoming traffic for the HPE LoadRunner Agent Process.

- When recording a .NET script on non-English operating systems with McAfee anti-virus active, it may issue the following message "The solution has been changed externally".

Workaround: Add the vugen.exe process to the Low-Risk processes in the McAfee antivirus On-Access Scan Properties.

- When you correlate a value from a snapshot, VuGen may create a boundary-based correlation, even though the recording correlation option is set to use regular expressions.

Additional Components

You can install additional components that provide advanced features for working with VuGen. The setup files are located in the **Additional Components** folder inside the root folder of the LoadRunner installation DVD or download folder.

Where to install additional components

The table below indicates which additional components are available, and where you should install each component:

Folder	Component	Description	Install on...
Agent for Citrix Server	SetupCitrixAgent.exe	<p>Installs the Citrix Agent which enhances VuGen's capabilities in identifying Citrix client objects during Citrix protocol record and replay. For installation instructions, see "Install the VuGen Citrix agent on the Citrix server. (Optional)" on page 437.</p> <p>The agent also enables you to use additional Citrix API functions. For details, see the Function Reference (Help > Function Reference).</p>	Citrix server
Agent for Microsoft Terminal Server	SetupMSTerminalAgent.exe	<p>Installs a utility that enhances the RDP protocol's recording mechanism in VuGen. For installation instructions, see "Installing the Microsoft Terminal Server Agent" on page 877.</p>	RDP server
Assembly Crawler for Analysis API	AssemblyCrawlerConsole.exe	<p>Installs a command-line utility to build a .NET configuration file for a LoadRunner Analysis API application. For more information, open the Analysis API Reference from the Start > Documentation menu on the LoadRunner machine (not available with VuGen Standalone).</p>	LoadRunner Analysis machine

Folder	Component	Description	Install on...
HostID Generator	Host ID Generator tool, licidgenerator.exe	Opens the Host ID Generator utility that displays the computer's Host ID. This is useful when requesting a license.	LoadRunner Controller machine
HPE NV (Network Virtualization)	<ul style="list-style-type: none"> NV4ControllerSetup.exe NV4LGSetup.exe 	NV4ControllerSetup.exe installs Network Virtualization for Controller, enabling emulation functionality. NV4LGSetup.exe installs Network Virtualization for the load generator machines and the NV Insights Report component for VuGen. For details, see the <i>NV Installation Guide</i> in the DVD's Additional Components/HPE NV folder.	LoadRunner Controller, VuGen, and load generator machines

Folder	Component	Description	Install on...
IDE Add-Ins	<ul style="list-style-type: none"> EclipseAddin \\hp.lr.vugeneclipse42addin.jar LRVS2013IDEAddInSetup.exe LRVS2015IDEAddInSetup.exe 	<p>Installs add-ins for supported versions of Visual Studio or Eclipse enabling you to create Vuser scripts in your standard development environment using the LoadRunner API. This integration also allows you to run the test directly from Visual Studio or Eclipse, to test its functionality.</p> <ul style="list-style-type: none"> Only 32-bit versions of Eclipse are supported. For more details on supported versions, see the System Requirements To install the Visual Studio Add-in, Visual Studio must be installed in the default location. For the LRVS2015IDEAddIn for Visual Studio, Visual C ++ language must be installed to work with C++ .Net Vuser projects. <p>For details, see "Create Scripts in External IDEs" on page 839.</p>	Visual Studio / Eclipse machine with VuGen

Folder	Component	Description	Install on...
IDE Add-Ins Dev	LREclipseIDEAddInDevSetup.exe LRVS2013IDEAddInDevSetup.exe LRVS2015IDEAddInDevSetup.exe	<p>Setup files for developer add-ins for supported versions of Visual Studio and Eclipse, enabling you to create NUnit or JUnit tests in your standard development environment using the LoadRunner API.</p> <ul style="list-style-type: none"> Only 32-bit versions of Eclipse are supported. For more details on supported versions, see the System Requirements To install the Add-in, Visual Studio must be installed in the default location. For the LRVS2015IDEAddIn for Visual Studio, Visual C++ language must be installed to work with C++ .Net Vuser projects. <p>For details, see "Create Scripts in External IDEs" on page 839</p>	Visual Studio or Eclipse machine with VuGen
LoadRunner ProtocolSDK	SetupLoadRunnerProtocolSDK.exe	Allows you to create and distribute custom LoadRunner protocols. For details, see " Protocol SDK " on page 875 .	Any machine with Virtual Studio 2015 and WiX Toolset 3.8 or higher
mobileRemote Agent	Select the relevant component for your operating system.	Enables you to capture a pcap file with Linux Redhat	

Folder	Component	Description	Install on...
SAP Tools	SapSpy.exe VerifyScripting.exe	<ul style="list-style-type: none"> • SAPGUI Spy. Examines the hierarchy of GUI Scripting objects, on open windows of SAPGUI Client for Windows. • SAPGUI Verify Scripting. Verifies that the SAPGUI Scripting API is enabled. For details, see "Configure the SAP Environment" on page 640 .	VuGen machine with SAPGUI client
Third Parties	Source files	The folder contains the source code of some third party software components which are being used in LoadRunner.	N/A
Virtual Table Server	SetupVTS.exe	Virtual Table Server (VTS) offers an alternative to standard LoadRunner parameterization. For details, see "Parameterization Overview" on page 342 .	Any machine

Standalone Applications

The following LoadRunner standalone applications are available in the **DVD/Standalone Applications** folder.

Folder	Component	Description	Install on...
Analysis Standalone	SetupAnalysis.exe	Installs LoadRunner Analysis as a standalone application. Install this to open LoadRunner results and create graphs and reports on a separate machine. For details, see Introducing Analysis .	Any machine
Load Generator	SetupLoadGenerator.exe	Installs the LoadRunner agent on the machine in order to run load tests. After you install this software, you access this machine from the Controller. For details, see Load Generators .	Any machine

Folder	Component	Description	Install on...
MI Listener	SetupMIListener.exe	Installs the HPE MI Listener, which servers as a router between the Controller and the LoadRunner agent. For details, see How to Set Up Your LoadRunner System Over Firewalls .	Dedicated machine
Monitors Over Firewall	SetupMoFW.exe	Installs the HPE Monitors Over Firewall component, allowing you to monitor servers located over a firewall. For details, see How to Set Up Your LoadRunner System Over Firewalls .	Dedicated machine
TruClient Standalone	SetupTruClient.exe	Installs TruClient as a standalone application. Install this tool to record Web applications with TruClient technology. You save the recordings to a script that can be used in a LoadRunner test run. For details, see the TruClient Help Center (select the relevant version).	Any machine
VuGen Standalone	SetupVuGen.exe	Installs LoadRunner Virtual User Generator (VuGen) as a standalone application, allowing you to create scripts for a load test. For details, see "Introducing VuGen" on page 34 .	Any machine

Protocol SDK

The Protocol SDK package allows you to create custom VuGen protocols from within Visual Studio.

VuGen provides this package as an extension to Visual Studio 2015.

Install the Protocol Library Package

1. **Prerequisite.** Make sure the following are installed:
 - Visual Studio 2015 with the Visual C++ language enabled
 - WiX toolset 3.10.3 to 3.11
 - WiX toolset add-in for Visual Studio 2015. If the WiX toolset was installed before Visual Studio 2015, you will need to reinstall it.
2. Locate the installation file, **SetupLoadRunnerProtocolSDK.exe**, on the LoadRunner DVD in the **Additional Components\LoadRunnerProtocolSDK** folder. You can install this extension on a machine that does not have an installation of LoadRunner.
3. Follow the installation wizard to completion.
4. Create a new test using the **LoadRunner Protocol SDK** template. For details, see the Protocol SDK

documentation, accessible from the Start menu or from the following (default) location:

C:\Program Files (x86)\HPE\LoadRunner\LoadRunner Protocol
SDK\documents\webframe.html.

Upgrade Protocol SDK Projects from Visual Studio 2012

The Protocol SDK is now an extension to Visual Studio 2015, and no longer an extension to Visual Studio 2012. Consequently, all projects created with LoadRunner 12.50 and earlier must be upgraded. For details, see **Upgrading to Visual Studio 2015** in the Protocol SDK documentation, available after installing the Protocol SDK.

For more details about the Protocol SDK, see the [LoadRunner knowledge base](#).

Installing the Virtual Table Server (VTS)

This topic describes how to install the Virtual Table Server (VTS). For information on using VTS, see "[Parameterization Overview](#)" on page 342.

Note: If you have an older version of VTS, uninstall it and then reinstall the latest version from the **Additional Components** folder.

To install VTS:

1. Run the setupVTS.exe file located in the **Additional Components\Virtual Table Server** folder in the installation media. The VTS Setup Wizard opens, displaying the welcome page.
2. Follow the online instructions to complete the VTS installation.
3. During the VTS installation process, the Configure VTS administration server screen appears. This screen lets you configure the VTS Administration server.
4. In the **Admin UI server port** box, keep the default value, 4000.
5. Click **Next** to continue with the installation. The Configure VTS screen appears.
6. Specify where to save the VTS data file.
7. Make sure that the **Start Virtual Table Server Automatically** check box is selected.
8. Click **Next**, and then follow the wizard's instruction to complete the VTS installation procedure.

Note: At the end of the installation process, a shortcut for VTS is created and added to the Start menu, **HPE Software > Tools > Virtual Table Server**. This shortcut gives you access to the VTS UI on the local machine. If you change the port that is used to access the VTS UI, you must manually update the URL property of the shortcut. For details on how to change the VTS UI access port, see **Configuring VTS** in the VTS online documentation.

If you are unable to access the VTS UI, make sure that the VTS Service service is started. To start the VTS Service service, go to **Control Panel > Systems & Security > Administration Tools > Services**. Right-click VTS Service and select **Start**.

See also:

- ["Parameterization Overview" on page 342](#) for details on when to use VTS

Installing the Microsoft Terminal Server Agent

Install the Agent for Microsoft Terminal Server

1. **Prerequisite:**

- Choose the RDP server machine on which you want to install the Microsoft terminal server agent. (Do not install the agent on a Load Generator machine.)
 - If you are upgrading, uninstall any earlier agent versions. See ["Uninstall the Agent for Microsoft Terminal Server" below](#).
2. If your server requires administrator permissions to install software, log in as an administrator to the server.
 3. Locate the installation file, **Setup.exe**, on the LoadRunner DVD in the **Additional Components\Agent for Microsoft Terminal Server** folder.
 4. Follow the installation wizard to completion.

Note: To use the agent, you must set the recording options before recording a Vuser script. In the Start Recording dialog box, click **Options**. In the Advanced Code Generation node, check **Use RDP Agent**.

Uninstall the Agent for Microsoft Terminal Server

1. If your server requires administrator privileges to remove software, log in as an administrator to the server.
2. Open **Add/Remove Programs** in the server machine's Control Panel. Select **HPE Software Agent for Microsoft Terminal Server** and click **Change/Remove**.

Troubleshooting and Limitations for Additional Components

Secure Channels

- You cannot use the Host Security Manager utility to update security settings on Linux load generators that use rsh (remote shell) to connect to the Controller.
- You cannot use the Host Security Manager utility to change the security mode of the load generator located over a firewall from off to on.
- When the load generator is located over a firewall, if the load generator and Controller have different security modes, communication cannot be established.
- If the Controller machine is using secure channel communication, the MI Listener should not be installed on the same machine as the Controller.

Function Reference

The HPE LoadRunner Function Reference describes functions that can be used in Vuser scripts in several HPE products. They can be used with supported protocols in scripts maintained in HPE Virtual User Generator for use in load testing, application management (HPE ALM), and functional testing (HPE Unified Functional Testing). For information about applicability, refer to the product documentation.

To open the Function Reference from a machine with LoadRunner installed, click **Start > All Programs > HPE Software > HPE LoadRunner > Documentation > Function Reference**. In icon-based desktops, such as Windows 8, search for **Function** and select **Function Reference** from the results.

Send Us Feedback



Let us know how we can improve your experience with the User Guide.

Send your email to: docteam@hpe.com